NASA CONTRACTOR REPORT 177366

INTELLIGENT REDUNDANT ACTUATION SYSTEM REQUIREMENTS AND PRELIMINARY SYSTEM DESIGN

P. DE FEO
L.J. GEIGER
J. HARRIS

CONTRACT NAS2-12081
SEPTEMBER 1985

NASA

NASA CONTRACTOR REPORT 177366

INTELLIGENT REDUNDANT ACTUATION SYSTEM REQUIREMENTS AND PRELIMINARY SYSTEM DESIGN

P. DE FEO
L.J. GEIGER
J. HARRIS

HR TEXTRON INC.
SYSTEMS ENGINEERING DIVISION
IRVINE, CALIFORNIA 92714

PREPARED FOR

AMES RESEARCH CENTER
UNDER CONTRACT NAS2-12081

SEPTEMBER 1985

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035

PREFACE

This document constitutes a report under Contract NAS2-12081, "Modular Microprocessor Techniques Development for Intelligent Actuation Management of Advanced Fly-By-Wire Systems." It represents the results of defining the requirements and conducting the preliminary system design and analysis. The work was conducted in HR Textron's Systems Engineering Division, Irvine, California. The Principal Investigator for this effort was Dr. Pio de Feo who was ably assisted by Messrs. James Harris, L.J. Geiger and Ed Buckley.

The NASA technical monitor for this task was Mr. K. C. Shih. The authors wish to acknowledge the guidance and helpful suggestions provided by Mr. Shih as well as the assistance of Mr. N. Rediess.

The material presented herein represents the findings of the authors and is not to be construed as being endorsed by the U.S. Government or the National Aeronautics and Space Administration.

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

TABLES

SUMMARY OF RESULTS

Some results of this analysis are only preliminary, and further study is needed to refine, confirm, and extend the scope of the findings. Some results and conclusions, however, are firm. The following is a list of the major conclusions:

IRAS CONFIGURATION

1. A minimum complexity, non-cross-strapped, quad configuration has been selected for the ISCU. This configuration meets or exceeds the reliability and fault tolerance requirements of the ISCU.

2. An Arbitrator is need to vote on channel status. It also performs the function of fault insertion and supports simulation control.

3. A communication path is provided among all DCPUs and the Arbitrator through a backplane common bus.

IRAS DYNAMICS

1. The digital IRAS simulation is in good agreement with Ames analog simulation.

2. The flow limiter has a significant effect on actuator dynamics in the case of high amplitude, high frequency inputs. The phase lag is 50 degrees, and the amplitude gain is -2.5 db. for a 5 Hz, 50% full scale input.

3. Digitized input commands from the FCC produce a phase lag which increases with the input frequency and the DCPU frame time. For a FCC frame time of 50 msec and a 3 Hz sine wave input, the phase lag is 21 deg.

4. A digital position loop closure does not produce significant time lags; however, significant overshoot characteristics develop which can be reduced by manipulating the forward loop gain. For a 50 msec FCC and a 5 msec DCPU frame time, the overshoot is 10% with a forward loop gain of 15.6 ma/v and less than 1% with a forward loop gain of 10 ma/v.

5. The digitized FCC input command significantly increases servovalve activity. The servovalve activity increases rapidly as a function of the DCPU frame time. For a FCC sample rate of 50 msec a DCPU sample rate of 5 msec and 1 Hz input sine wave, the normalized power spectra peak is 9%.

6. Notch filters are very effective in damping the servovalve activity; however, they add to the phase lag which increases as a function of the DCPU frame time. For the previous case, the power spectra is reduced from 9% to .2%, and an additional phase lag of 10 degrees is introduced.

IRAS HARDWARE ARCHITECTURE

1. The DCPU, EC and Arbitrator will be emulated in 68000 microprocessors.

2. The TCT will be implemented in a IBM PC/AT.

3. The architecture of the CIU is a function of the position loop closure, digital or analog.

4. Although no absolute requirements have been established for a FCC/DCPU bus structure, none of the examined protocols is well suited to support that communication path.

5. All communications to/from the IBM PC/AT are implemented via RS-232 point-to-point serial links.

IRAS SOFTWARE ARCHITECTURE

1. A VAX hosted software development environment will be used for IRAS embedded software.

2. The "C" language will be the HOL for IRAS embedded software.

INTRODUCTION

Advanced Flight Control Systems (AFCS) have very high operational capabilities and very high reliability, availability and survivability requirements. Several redundant configurations of actuation systems meeting those requirements have been designed and successfully demonstrated. However, this has been accomplished with extensive duplication of hardware components (brute force redundancy) and has resulted in significantly increased computational loads of the Flight Control Computers (FCC) which perform the failure analysis and the reconfiguration management.

Modern advanced technology has made it possible to produce powerful microprocessors at low cost featuring low volume, weight, and power requirements. These microprocessors can effectively perform locally, at the actuator level, the tasks of failure isolation and configuration management, which were previously performed by the FCC, and other related tasks. This new actuation system concept called an Intelligent Redundant Actuation System (IRAS) significantly reduces the FCC computational requirements and additionally provides the capability to perform those local tasks in a more advanced and comprehensive manner than previously feasible. This ultimately results in increased reliability, availability, survivability, and maintainability and lower life cycle costs for the total AFCS.

The purpose of this document is:

1. to outline the requirements for a flexible IRAS experimental breadboard system capable of demonstrating the feasibility of the concept and of exploring a variety of configurations and,

2. to provide a data base for future research activities in the IRAS area.

SYSTEMS DESCRIPTIONS

A schematic of IRAS is shown in Figure 1.

The major components are:

1. The Emulation Computer (EC)

   The EC includes the Emulated Diagnostic and Maintenance Computer (EDMC) and the Emulated Flight Control Computer (EFCC). The EDMC performs diagnostics and maintenance tasks during flight and on the ground; and AI concepts and structures can be used which effectively support these functions. The EFCC emulates configurations of Flight Control Computers and hosts the vehicle dynamics to provide a closed loop, real-time analysis environment. The EDMC and EFCC are linked to the Intelligent Servoactuator Control Unit (ISCU) and are used to drive the Digital Pilot Display (DPD).

2. Intelligent Servoactuator Control Unit (ISCU)

   The ISCU consists of the Digital Command Processing Unit (DCPU) and the Control Interface Unit (CIU). The microprocessor-based DCPU provides the computational and logic capabilities for calculating the actuator coil commands; for monitoring the actuator status; and, for implementing reconfiguration management strategies as a function of detected failures, and the vehicle state and control mode. Since the DCPU is the key component of IRAS, most of this document relates to it. The analog CIU provides the interface between the DCPU and the actuator coils, LVDT and solenoid. The current loop closure (inner loop) is implemented within the CIU. The position loop (outer loop) can be implemented within the CIU or within the DCPU. The two implementations are discussed in detail later in this document. A key requirement of the ISCU is that it be very powerful and flexible so that the concept feasability can be demonstrated and implementation issues of adaptive digital control techniques applicable to a variety of servoactuator and FCC configurations may be analyzed. Futhermore, the ISCU, when operating within the dynamic environment of the EC, is a powerful test-bed for analyzing the effects of advanced gain optimization and limiting techniques on the safety and performance of fixed and rotary wing vehicles.

3. Digital Pilot Display (DPD)

   The DPD is a multipurpose display which provides the

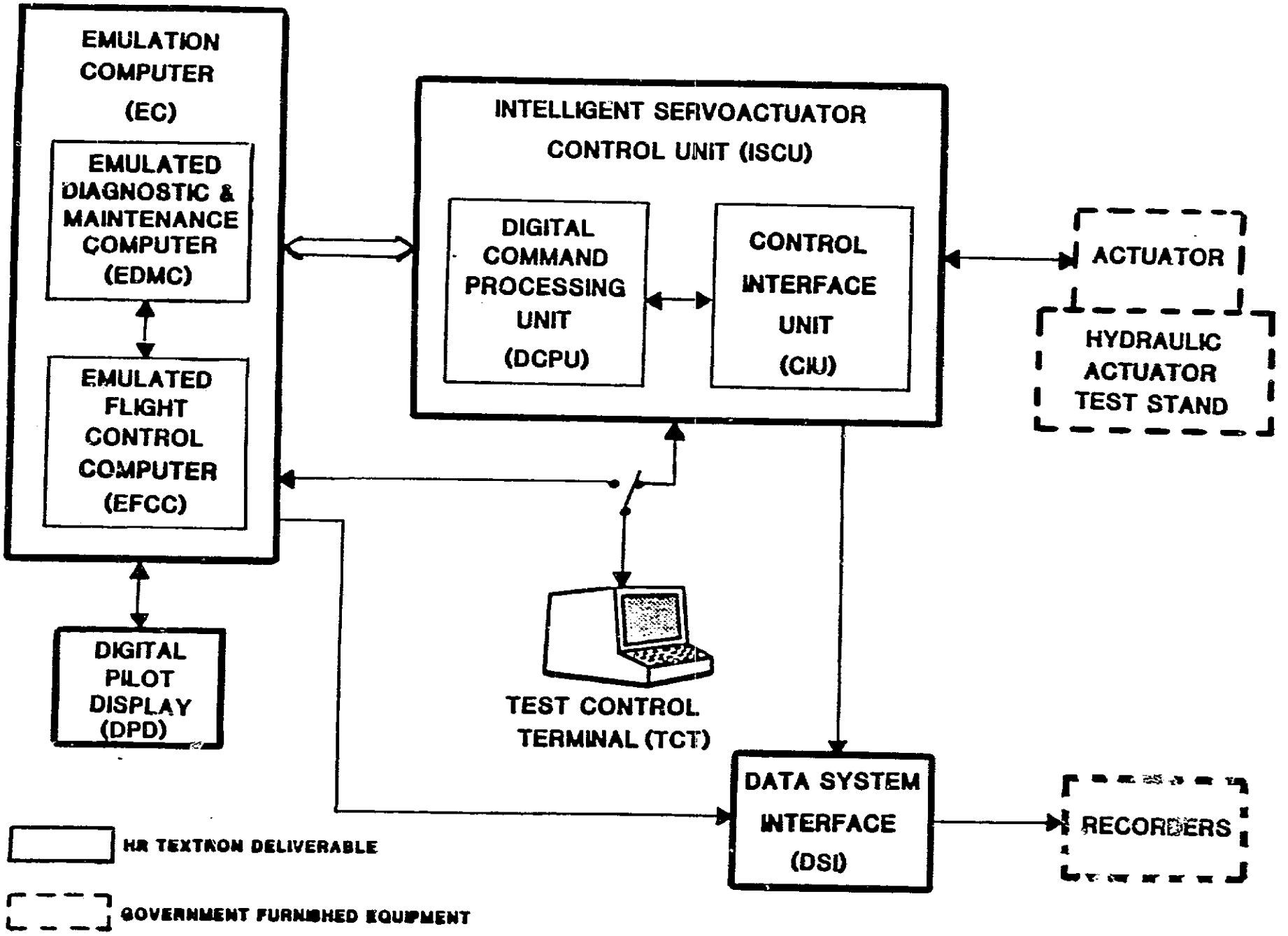


FIGURE 1 DIAGRAM OF IRAS SYSTEM

pilot with information and interfaces needed to control and monitor the actuation system among other things. It is driven by the EFCC and its functions are implemented within the Test Control Terminal capability.

4. Test Control Terminal (TCT).

The TCT's major functions are: (1) to control the operation of the IRAS; and, (2) to provide a work station for software development. It also provides a realistic way to emulate the DPD function of the pilot interface with the IRAS. A commercial personal computer (PC) can provide all the capabilities needed to support these functions.

5. Data System Interface (DSI)

The DSI provides the capability of interfacing the EC and DCPU to analog instrumentation such as strip chart recorders. It consists of standard Digital to Analog converters with appropriate characteristics.

The IRAS lab system, (referred to as IRAS throughout this document,) and the equipment currently available at NASA Ames provide a versatile integrated environment to implement and analyze a wide variety of IRAS configurations. Major issues which can be explored are:

Redundant Actuator Monitor and Reconfiguration Strategies

Redundancy Actuator Control Configurations

Interfaces to the Actuator and FCC

Maintenance and Diagnostic Strategies

Cockpit Displays

The IRAS must be very flexible and easily configured to support a variety of FCC actuators and actuator controller configurations. Initially the IRAS is being configured to interface with a quad, synchronized FCC configuration and the TAFCOS Actuator. The TAFCOS Actuator, currently operational at NASA Ames, was developed by HR Textron under NASA Contract NAS2-11512.

Two areas with significant technical promise which can be investigated using the IRAS are: (1) the use of Artificial Intelligence (AI) concepts for diagnosis and maintenance purposes; and, (2) the implementation of a digital position loop closure for the servoactuator. These and other research areas are discussed in detail in other sections of this document.

SYSTEM REQUIREMENTS

The IRAS design requirements are grouped as follows:

- o System Architecture Design Requirements
- o Reliability and Fault Tolerance Requirements
- o Failure Detection and Reconfiguration Requirements
- o Dynamic Performance Requirements

The three major high level system requirements are:

1. The system must be capable of interfacing with a quad redundant, synchronous FCC, and a four coil, flux summing, redundant TAFCOS actuator. The IRAS architecture must also be very flexible so that a wide variety of FCCs and actuation system configurations can be supported and analyzed.

2. The reliability and fault tolerance of the Intelligent Servoactuator Control Unit (ISCU) must be consistent with the reliability and fault tolerance of the FCC and TAFCOS actuator.

3. The overall dynamic characteristics of the TAFCOS actuation system should not change significantly when the analog servocontroller is replaced by the digital ISCU.

While all the requirements impact on all the elements of the IRAS, the only significant effect of interest in the current study is on the ISCU configuration because:

1. The Test Control Terminal (TCT) performs only secondary non-critical tasks relative to the actual control of the actuation system.

2. The issues related to the reliability and fault tolerance of the Emulated Diagnostic and Maintenance Computer (EDMC) and of the Digital Pilot Display (DPD) are not a research objective of this effort, at least at the present time.

3. The Emulated Flight Control Computer (EFCC) is included in the IRAS only to provide the dynamic environment needed for analyzing the performance of the actuation system and not for developing and analyzing FCC configurations.

SYSTEM ARCHITECTURE DESIGN REQUIREMENTS

The IRAS system architecture design requirements primarily effect the architectural design of the ISCU, which is the central element of the IRAS and is the focus of the research objectives of this effort. The architecture of the ISCU must provide a level of reliability and fault tolerance which is comparable to the one existing for the FCC and for the most redundant elements of the actuator and servovalve coils.

The FCC and the servovalve coils have a quad configuration which provides a Fail-Op^2/Fail-Safe performance. Therefore, the most natural candidate configuration for the ISCU is also a quad configuration which can be easily interfaced with the FCCs and actuator.

Quad configurations can be easily modified into dual, and dual-dual configurations. Furthermore, they typically include all the major hardware components needed to implement triplex configurations. However, conversion to triplex configurations might require extensive software redesign especially in the areas of voting and signal selection; and design and fabrication of additional interfaces. The only configuration which cannot be easily implemented starting from a quad configuration is the very powerful, dual-triplex configuration, such as the Primary Flight Control System of the ADOCS. In fact, that configuration requires a set of six microprocessors, two more processors than quad configurations. The uniform redundancy existing throughout the channels of the IRAS quad configuration is shown in Figure 2.

A key functional requirement of the IRAS is the capability to isolate a failed channel. It is not safe to delegate that responsibility to the failed channel itself because the fault might prevent the failed channel from reliably performing that critical task. Therefore, the responsibility of channel isolation must be delegated, and shared by, the remaining operational channels. This approach requires the implementation of a device, external to the four ISCU channels, which performs the critical role of arbitration. The Arbitrator is a single point failure in the system and, therefore, in a flight critical environment, must be a highly reliable, fault-tolerant device. In the IRAS environment, the functions of the Arbitrator can be emulated by a separate microprocessor unit which interfaces directly to the DCPU and the CIU. This configuration is very appropriate because: (1) in the IRAS lab environment, the issues of criticality are addressed from a functional and structural point of view only and not from an implementation point of view; and, (2) the IRAS is required to be very flexible so that different architectures and algorithms can be developed and analyzed.

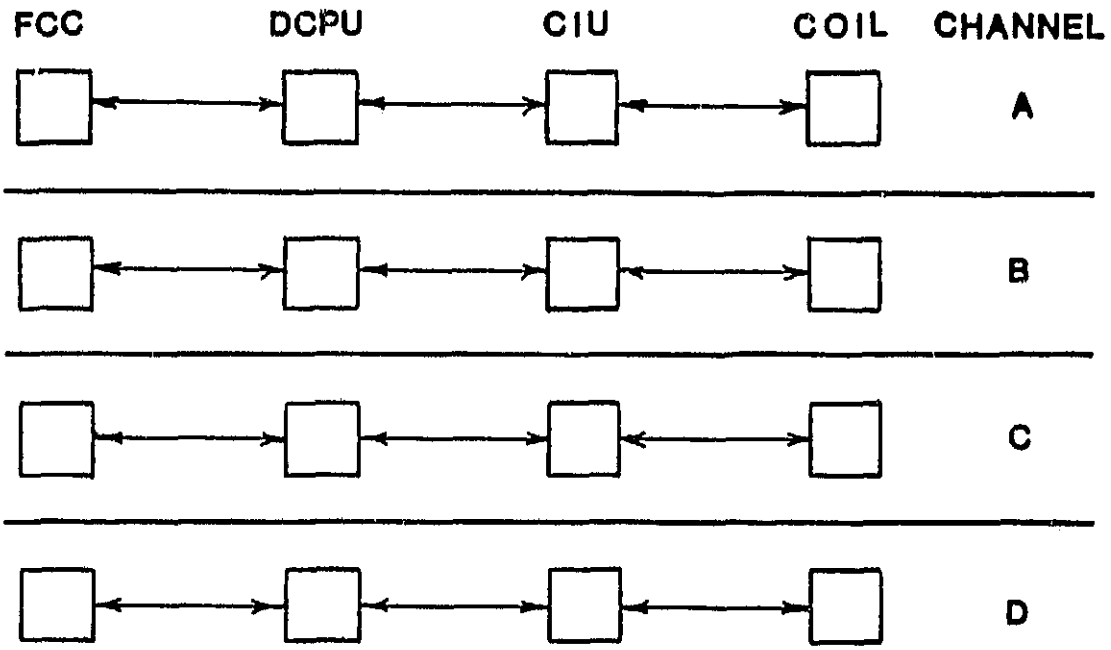


FIGURE 2 IRAS QUAD CONFIGURATION

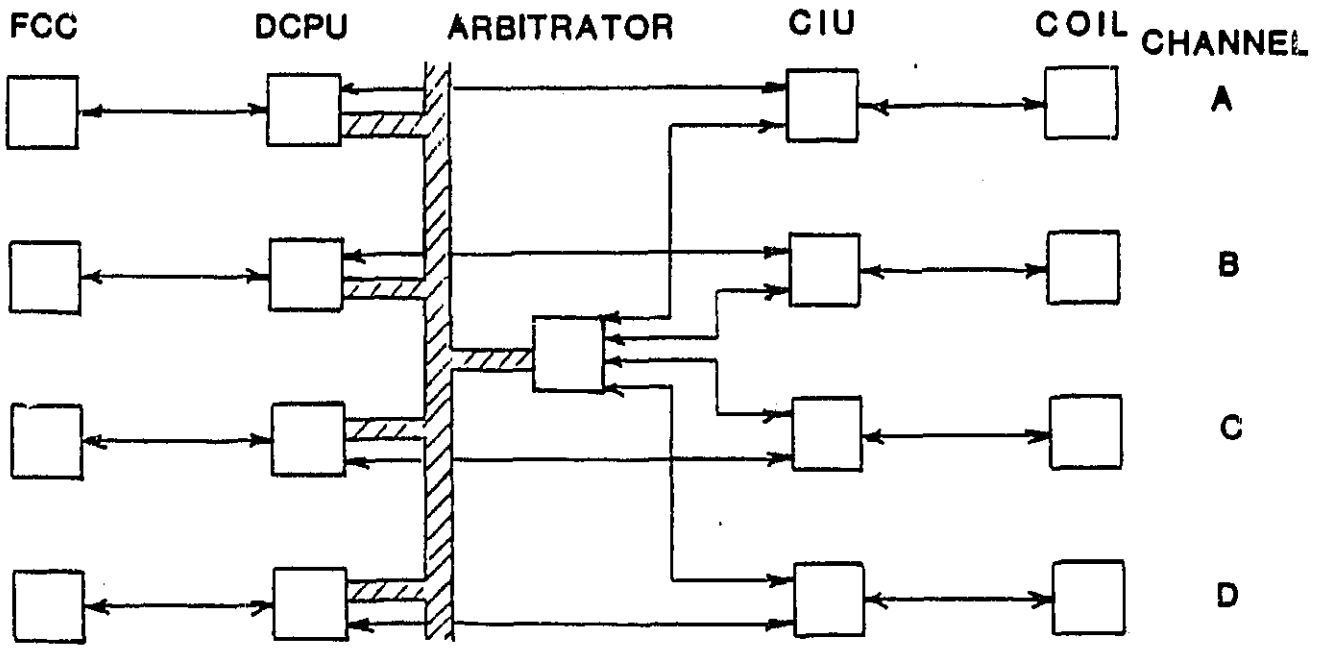


FIGURE 3 IRAS QUAD CONFIGURATION WITH ARBITRATOR

A configuration of the IRAS which includes the Arbitrator is shown in Figure 3. The major characteristics of this configuration are:

1. Communications among channels are minimized (no cross-strapping) which reduces design complexity and parts count.

2. Communications between corresponding FCC and DCPU pairs are shown as serial or parallel digital dedicated links. The capability to implement or emulate a bus architecture to link the FCCs and DCPUs is also provided. Issues related to the choice of appropriate bus protocols are discussed later in this document.

3. A communication link which can be implemented through a backplane common bus connection is shown between the DCPUs. The IRAS provides that simplex bus interface to support the configuration flexibility requirement. The capability to emulate a redundant bus structure is also provided so that related issues can be investigated.

   This communication path provides the capability of establishing voting planes within the DCPUs at the input level, at the output level, or at any intermediate computational level. The path is simplex in the IRAS, but if it supports critical functions in a flight worthy implementation, it must be redundant.

4. The communications between DCPU and CIU pairs are shown as a serial or parallel dedicated links. A digital link via the DCPU internal bus is also possible.. The optimum configuration is a function of whether the position loop (outer-loop) closure is performed in the digital DCPU or the analog CIU. These issues are analyzed later. The IRAS has the capability of implementing or emulating a variety of DCPU/CIU communications, including analog communications and the communications via the DCPU internal bus.

5. The Arbitrator is emulated in a microprocessor identical to that used for the DCPUs. It can be located in the same DCPU enclosure to facilitate communication with the DCPUs through the common backplane connections. The Arbitrator also communicates with the CIUs which control the relays to isolate failed channels.

RELIABILITY AND FAULT TOLERANCE REQUIREMENTS

The key feature of fault tolerant systems is the capability of failure detection and isolation, and subsequent reconfiguration. Inter-channel comparison monitoring and intra-channel self-test techniques are used for this purpose.

Self-test techniques can only achieve limited coverage, less than 100% when applied to complex digital equipment such as the DCPU. Furthermore, a significant delay can exist between the time of fault activation and the time of fault detection. This is due to the long execution time often required by self-test algorithms and by the low execution priority of these algorithms in time-critical environments. However, self-test techniques do not require an increase in hardware resources or hardware complexity, because they are usually entirely implemented in software. Self-test is often used, in a background mode, to isolate failed components in a faulted channel and to select the non-failed channel between the last remaining two, with a probability better than 50% (and less than 100%). Self-test is also used to determine the availability of components to perform future tasks. Examples are preflight tests; and in-flight tests to determine compliance with the requirements of equipment availability prior to engaging critical functions, such as low visibility automatic landing.

On the other hand, comparison monitoring has the capability to rapidly and exhaustively detect and isolate faults, if a minimum inventory of non-failed components is available. Comparison monitoring is easily implemented in synchronized configurations such as the IRAS. In these configurations, the thresholds of the comparitors can be set to rather small values for early fault detection while still minimizing the danger of nuisance disconnects. However, comparison monitoring techniques require hardware duplication, the implementation of voters, and increased complexity of the interfaces not otherwise needed. Comparison monitoring is primarily used in critical applications to rapidly remove faulty components and prevent propagation of errors throughout the system.

The IRAS structure provides the capability of implementing self-test and comparative monitor techniques. A promising set will be implemented and analyzed. It will allow the inclusion of additional techniques to extend fault coverage, to decrease detection time, and to more effectively utilize the inventory of non-failed equipment, so that under any conditions, the maximum number of operational channels is maintained. The initial set of failure detection techniques to be implemented in IRAS is discussed later.

The overall reliability and fault tolerance of a redundant configuration is a function of four major factors:

1. The coverage of the fault detection techniques used.

2. The reliability of each component.

3. The level of redundancy provided for each critical functional component, and

4. The level of cross-strapping existing among components.

Factors three and four establish what reconfiguration capabilities are available after the failure of one or more components. A reliability analysis has been performed on the basic IRAS architecture shown in Figure 2 to determine the relative merits of three basic configurations which utilize the same number and type of components but have different interfaces and reconfiguration management approaches. These three configurations, outlined in Figure 4, are:

Configuration "A". Quad configuration. No cross-strapping among channels

Configuration "B". Quad configuration. Cross-strapping between DCPU and CIU.

Configuration "C". Dual-dual configuration.

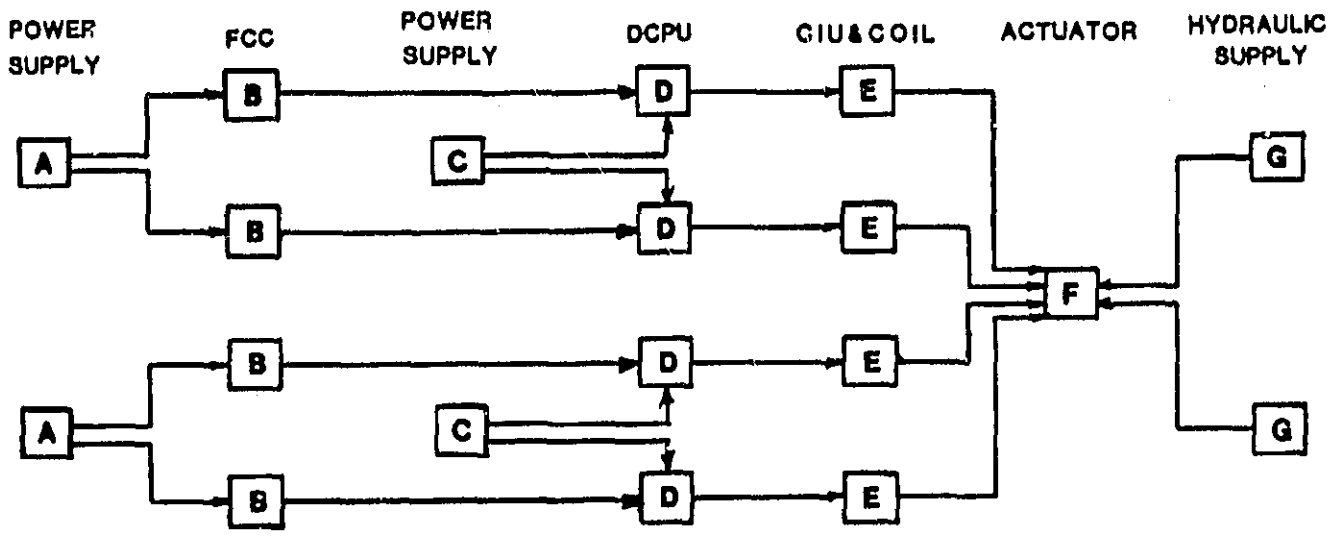
The dual-dual configuration provides only a Fail-Op/Fail-Safe operation, while both quad configurations provide a Fail-Op/Fail-Op/Fail-Safe operation which is consistent with the FCC and the TAFCOS actuator. A major functional difference between the two quad configurations is that in configuration "A" the loss of any FCCs or DCPUs implies the loss of the corresponding actuator coils. Therefore, if two DCPUs fail, the performance of the actuator is somewhat degraded even if none of the coils have actually failed. In the case of configuration "B", the performance of the actuator remains unchanged even if two DCPUs fail, provided that none of the coils have failed. In fact, in this case each coil can receive valid inputs from the remaining operational DCPU channels.

The following assumptions have been made in the reliability analysis:

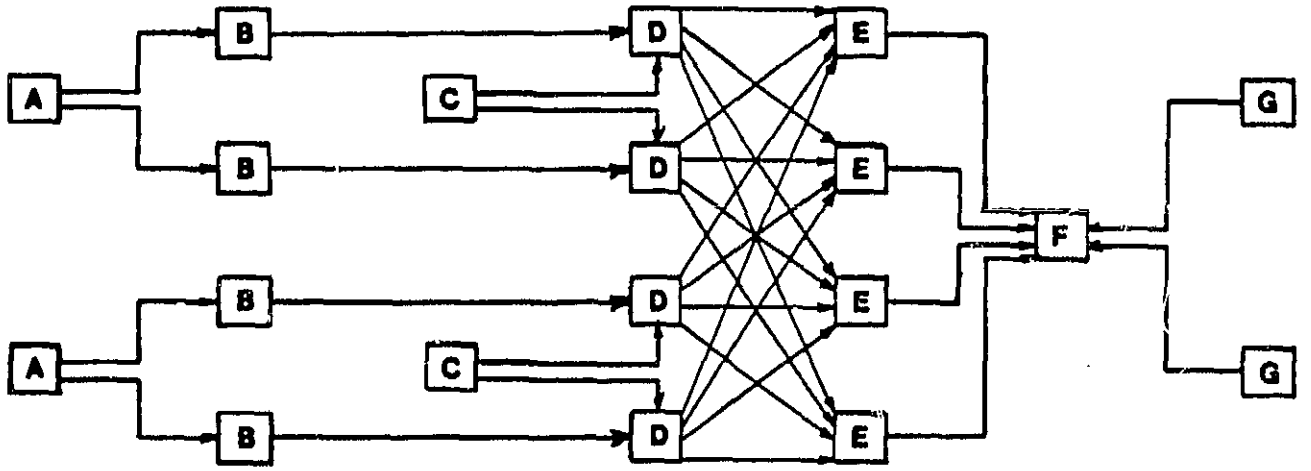
- Separate dual power supplies are available for the FCCs and DCPUs.

- Dual hydraulic lines provide the hydraulic power to the actuator cylinder.

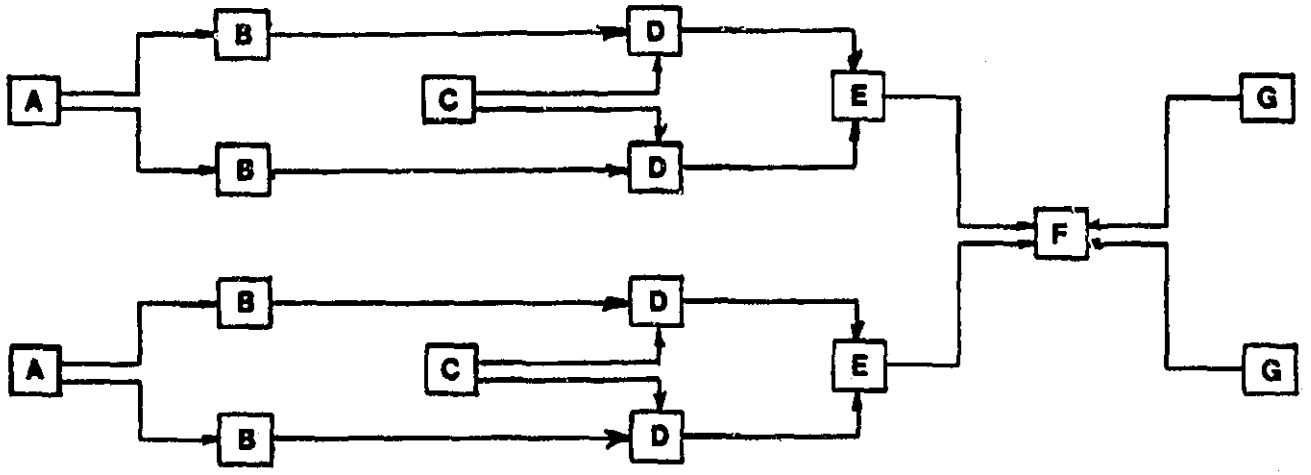
- The voters and the Arbitrator have zero failure rates (voters and Arbitrator are not shown in the diagram).



CONFIG. A - QUAD WITH NO CROSS-STRAPPING



CONFIG. B - QUAD WITH DCPU/CIU CROSS-STRAPPED



CONFIG. C - DUAL-DUAL

FIGURE 4 RELIABILITY DIAGRAMS - IRAS CONFIGURATIONS

- o In all configurations, corresponding components have same the failure rates.

- o The failure rate of the actuator is typical of redundant configurations.

The reliability diagrams of configuration "A", "B" and "C" are shown in Figure 4. The diagrams clearly show the dependency existing among stages. The loss of either of the two electrical power supplies results in the loss of the two corresponding channels, except in the case of the cross-strapped "B" configuration. The loss of any component to the left of the actuator, other than the electrical power supply, fails one entire channel in configuration "A". By contrast, the cross-strapping existing between the DCPU and the CIU/coil combination in configuration "B" prevents failure of any components to the left of the CIU/coil from failing the entire channel. In configuration "C", the loss of any component to the left of the CIU/coil fails one entire pair of channels.

The component failure rates in all three configurations are shown in Table 1. The failure rate assumed for the actuator is $.01 \times 10^{-6}$ Failures/Hour which is too low for the simplex configuration of the TAFCOS actuator. The value was chosen because it is representative of redundant actuator configurations for critical applications. The failure rate of the other IRAS components has been estimated based upon the reference noted in Table 1. It should be noted that the purpose of this preliminary analysis is to access the relative merit of each IRAS configuration rather than provide an accurate failure rate of each configuration. In the context, the results of this analysis do not vary significantly with small changes in component failure rates, although large changes can effect the conclusions. Efforts are being made to acquire more accurate component failure rates which will increase the accuracy of the absolute reliability estimate of each of the three IRAS configurations.

The results of the reliability analysis are summarized in Table 2. The first horizontal row of numbers represents the total reliability of each configuration. The results show that the dual-dual configuration has a failure rate equal to 5.2×10^{-7} Failures/Hour, fifteen times higher than the most reliable configuration, the quad cross-strapped configuration (3.64×10^{-8} Failures/Hour). The quad configuration with no cross-strapping between the DCPU and the CIU/coil has a failure rate of 4.08×10^{-8} Failures/Hour, and it is only 15% more than the cross-strapped configuration. Therefore, a preliminary conclusion indicates that the rather modest decline of the predicted failure rate does not justify the additional complexity of cross-strapping.

The other data in Table 2 represents the percentage change of the total failure rate as a result of doubling each component failure rate. As an example, if the actuator rate is doubled

TABLE 1 COMPONENT FAILURE RATES (FAILURES/HOUR)

LABEL	COMPONENT	FAILURE RATE ($\times 10^{-6}$)
A	FCC POWER SUPPLY	4
B	FCC	145
C	DCPU POWER SUPPLY	4
D	DCPU	145
E	CIU&COIL	200
F	ACTUATOR	.01
G	HYDRAULIC SUPPLY	130

NOTE: FAILURE RATES ARE BASED ON THE STUDY "ASSESSMENT OF ADVANCED FLIGHT CONTROL SYSTEMS RELIABILITY AND MAINTAINABILITY DESIGN, QUALIFICATION, AND MAINTENANCE REQUIREMENTS" T. BRADY - USAAVRADCOM-TR-82-D-MAY 1982

TABLE 2 CONFIGURATION FAILURE RATES AND PERCENT CHANGE FOR 100% INCREASE IN COMPONENT FAILURE RATES

	CONFIG. A QUAD NO CROSS-STRAP	CONFIG. B QUAD CROSS-STRAP	CONFIG. C DUAL-DUAL
TOTAL FAILURE RATE (FAILURE /HOUR)	4.08×10^{-8}	3.64×10^{-8}	5.2×10^{-7}
COMPONENT (F.R. $\times 10^{-6}$)			
ELECTRICAL POWER SUPPLIES (4)	32%	26%	14%
FCC (145)	7	13	48
DCPU (145)	13	13	86
CIU & COILS (200)	16	1	69
ACTUATOR (.01)	25	27	6
HYDRAULIC POWER (130)	87	98	29

from the initial value of .01 x 10^{-6} Failures/Hour to the value of .02 x 10^{-6} Failures/Hour, the corresponding failure rate of the two quad configurations increases by 25% and 27%. An interesting result of this analysis is that the sensitivity of CIU/coil failure rate is very small (1%) in the cross-strapped quad configuration, while it is substantial (18%) in the case of the quad configuration with no cross-strapping. The reason is that, in the case of the cross-strapped quad configuration, a failure of one CIU/coil component does not propagate to the left, leaving the integrity of the FCC and DCPU unchanged. In the non cross-strapped quad, the loss of a CIU/coil fails the entire channel depleting the inventory of available spare components.

Although the results of this reliability analysis are rather preliminary, they indicate that a simple straight forward quad configuration with minimum interchannel interfaces provides the required reliability and the required fault tolerance. This is the IRAS configuration that will be implemented first.

FAILURE DETECTION AND RECONFIGURATION REQUIREMENTS

All the IRAS diagnostic functions are performed within the DCPUs. They include all the functions previously performed within the TAFCOS analog Electronic Servoactuator Control Unit (ESCU) and additional functions which are primarily related to the diagnostics of the DCPUs themselves. In each area, the diagnostic capability of the DCPUs are only limited by the availability of appropriate feedback variables, rather than by the DCPU internal computational power. This is particularly true in the case of the actuator set which in the current configuration has a very limited number of health monitors.

While some of the diagnostic functions can be defined only after the design of the IRAS is completed, the following functions will be most certainly included in any design:

1. A comparison monitor of each coil current with the corresponding command value. This test detects failure in the servovalve and the CIU electronics.

2. A comparison monitor of the demodulated LVDT position feedback with the output of a software implemented, servoactuator model driven by the same command. This test detects servoactuator and LVDT failures.

3. A test of the power supplies.

4. DCPU cross-channel comparison monitoring. The capability exists to implement cross-channel voting strategies at the DCPU input plane, DCPU output plane, or any appropriate intermediate computational planes. The capability is provided with a backplane communication link among all DCPUs.

5. DCPU self diagnostics.

6. Wrap-around tests.

Test strategies 1, 2 and 3 are executed by each DCPU every frame time on an intrachannel test basis only, (i.e., in each channel the DCPU only monitors equipment in its own channel). This is consistent with the quad, low complexity non cross-strapped configuration of the IRAS. The implication of these tests is that if any of them fail, the channel is disengaged. In some cases, cross-strapping would allow the disengagement of the failed component only, instead of the entire channel. As an example, if one CIU is detected failed then the remaining operational CIUs could provide the current input to the coil in the channel of the failed CIU, if the CIUs and the coils were cross-strapped.

Test strategy 4 provides several capabilities. A voting plane at the DCPU output provides a catch-all failure detection mechanism for DCPU and FCC failures. This is an interchannel test in which every operational channel compares its output with the output of the remaining operational channels to determine if there is agreement or disagreement within specified thresholds. One additional voting plane, provided by the Arbitrator, might be necessary to resolve disagreements among channels relative to the status of another channel. Task 4 can also be applied at the DCPU input voting plane to mask FCC failures.

Test strategy 5 is an intrachannel DCPU test strategy in which the test sequence is organized according to the criticality of the function to be tested. An example of this test sequence follows. The first step initiated by the EDMC is the test of the CPU instruction fetch and execution sequences. The next step is the test of the DCPU power supply. A monitor card is needed to support this test. Next the program memory and data memory are tested. The program memory can be tested using signature analysis methods; the data memory can be tested by forcing each code memory cell to the values of 1 and 0. The last step of the test sequence is a test of the DCPU real-time clock. This test can either use watch-dog timing loops or the clocks of the other DCPUs. The timing loop test consists of the execution of pretimed fixed sequences of instructions. A fault of the CPU or clock is detected by a difference which exceeds a threshold value between the actual execution time and the prestored value. The real-time clock can be also tested by forcing a rendezvous among all DCPUs. A fault is detected whenever one DCPU is too early or too late relative to the other processors at rendezvous time.

Wrap-around techniques are very powerful for testing interfaces, I/O ports and converters. Test strategies 1 and 2 when applied to coil currents and LVDT signals are examples of this technique. Wrap-around can also be applied to: (1) the communications between EFCCs and DCPUs, and between DCPUs and CIUs; and, (2) all the converters within the DCPU. Within the IRAS, wrap-around techniques will be extensively used whenever feasible.

All the tests which have been described can be executed during pre-flight and post-flight maintenance and diagnostic procedures without limitations. The same tests with appropriate modifications can also be used for in-flight failure detection and isolation, but special considerations must be observed to avoid conflict with the execution of the critical real-time control algorithms. Within an active control channel, tests of the CPU instruction set, the program memory and the power supply can be executed during flight in a background mode within an active control channel. The communica-tions between the EFCC and the DCPU can also be tested provided there is sufficient additional bandwidth in the transmission link. The data memory test cannot be executed in an active channel in flight, because of the destructive nature of the test. However, the test can be executed in a failed channel, as a part of the failure identification procedure.

One powerful feature of the IRAS is the capability of reconfiguring itself after one or more faults. Basically the reconfiguration strategy is the following:

1. With no failed channel, every channel bears an equal load and every actuator coil is energized to the same level.

2. After the first failure, the failed channel is isolated and a gain adjustment is made in the remaining three channels to compensate for the loss.

3. After the second failure, the second failed channel is also isolated and another gain adjustment is made in the remaining two operational channels to compensate for the loss of the second failed channel.

4. After the third failure, no attempt is made to control the servoactuator which is de-energized by removing its hydraulic power.

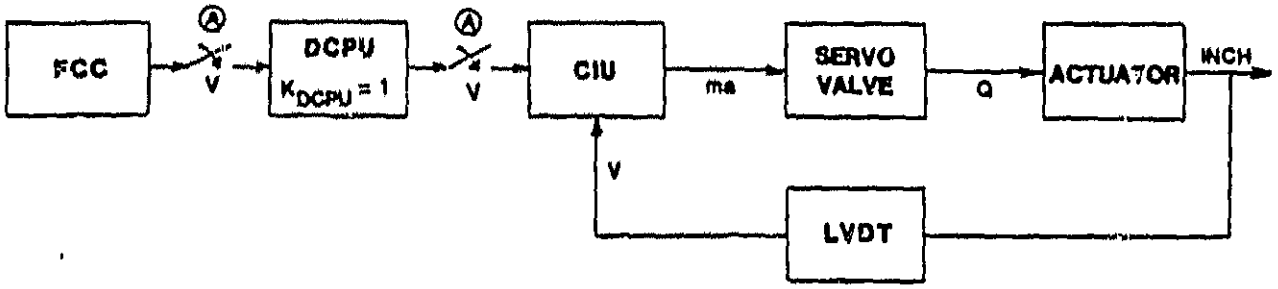
Several strategies have been considered for performing these reconfiguration tasks which lead to different IRAS configurations. The strategies and corresponding configurations are described in another section of this document.

DYNAMIC PERFORMANCE REQUIREMENTS

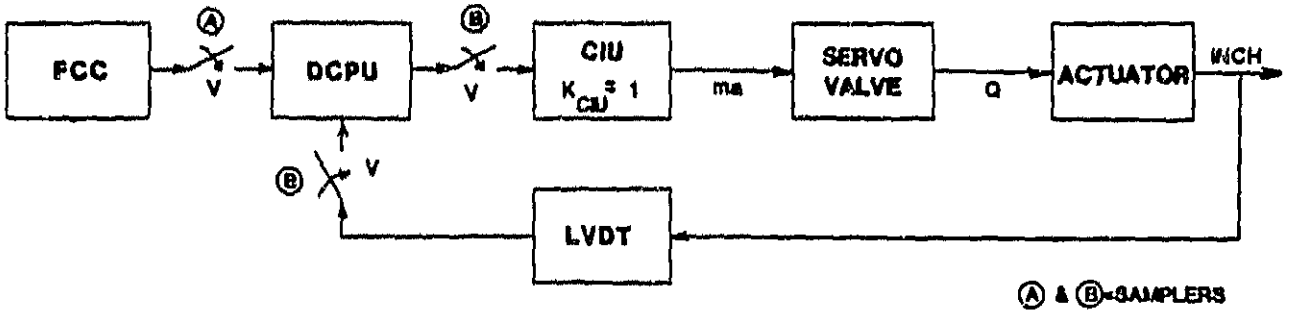
In this section of the report the dynamic characteristics of several IRAS configurations are analyzed. To support this study, a digital simulation was developed consistent with the analog simulation at AMES. A functional diagram of the simulation and a table of constants are shown in Figure 5.

The simulation was configured to represent:

1. An analog FCC structure and an analog IRAS position loop control configuration.



POSITION LOOP CLOSURE - ANALOG



POSITION LOOP CLOSURE - DIGITAL

COMPONENT	TRANSFER FUNCTION	VALUES
DCPU/CIU	$K_{DCPU} \cdot K_{CIU} \Big\vert_{LIMITED}$	$K_{DCPU} \cdot K_{CIU} = 15.6 ma/v$ LIMITER = ± 32ma
SERVOVALVE	$\frac{K_{SV}}{(S/\omega_{SA}+1)(S/\omega_{SV}+1)}\Big\vert_{LIMITED}$	$K_{SV} = .0351$ CIS/ma $\omega_{SA} = 1000$ RAD/SEC $\omega_{SV} = 440$ RAD/SEC LIMITER = ±.561 CIS
ACTUATOR	$\frac{1}{S} \quad \frac{K_A \omega_A^2}{S^2 + 2\zeta_A \omega_A S + \omega_A^2}\Big\vert_{LIMITED}$	$K_A = 6.578$ IN-SEC^{-1}/CIS $\zeta = .036$ $\omega_A = 3608$ RAD/SEC LIMITER=±.45 INCH
LVDT	K_{LVDT}	$K_{LVDT} = 22.22$ VOLT/IN

FIGURE 5 DIGITAL SIMULATION FUNCTIONAL DIAGRAM

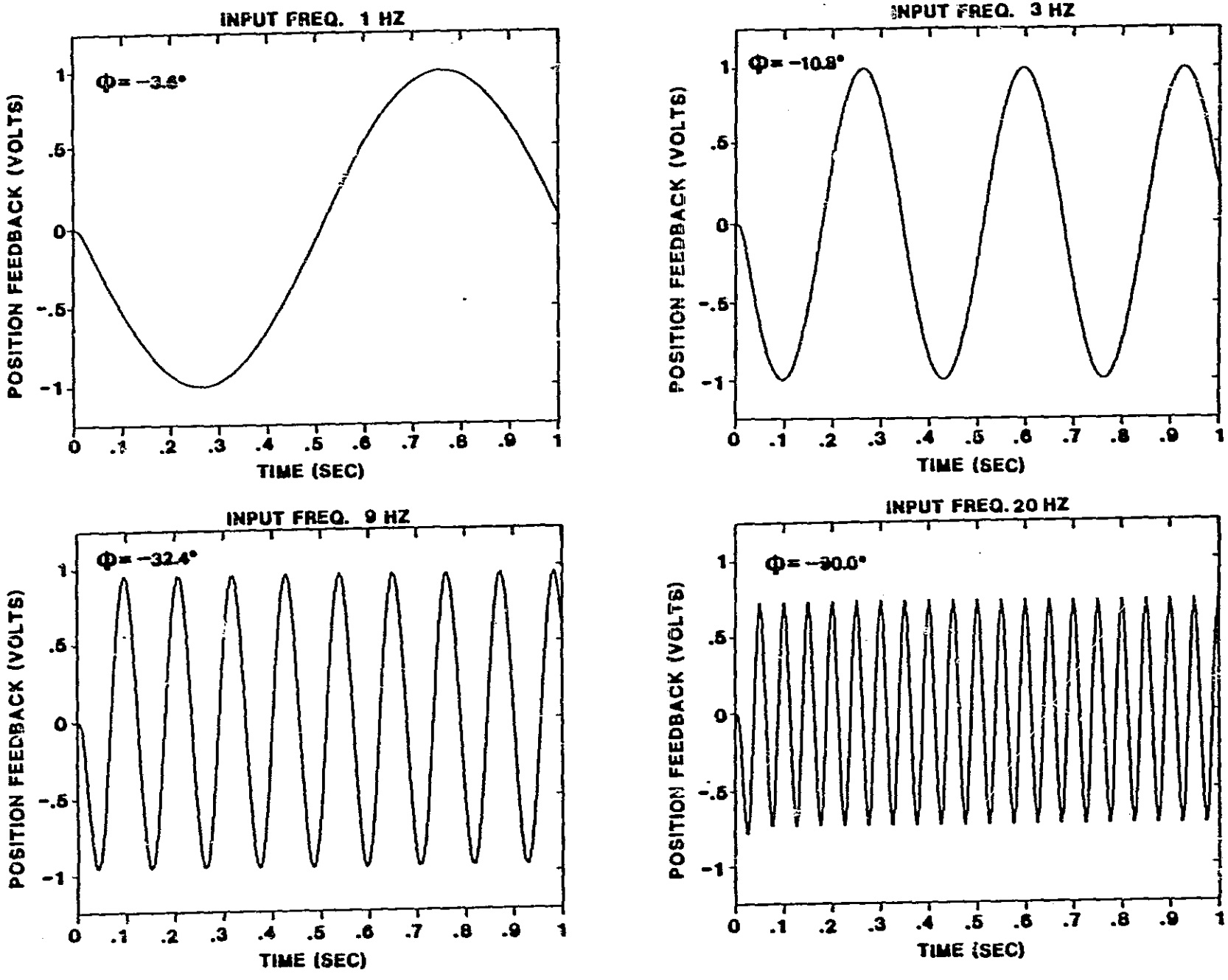


FIGURE 6 FREQUENCY RESPONSE, ANALOG INPUT & IMPLEMENTATION 10% FULL SCALE

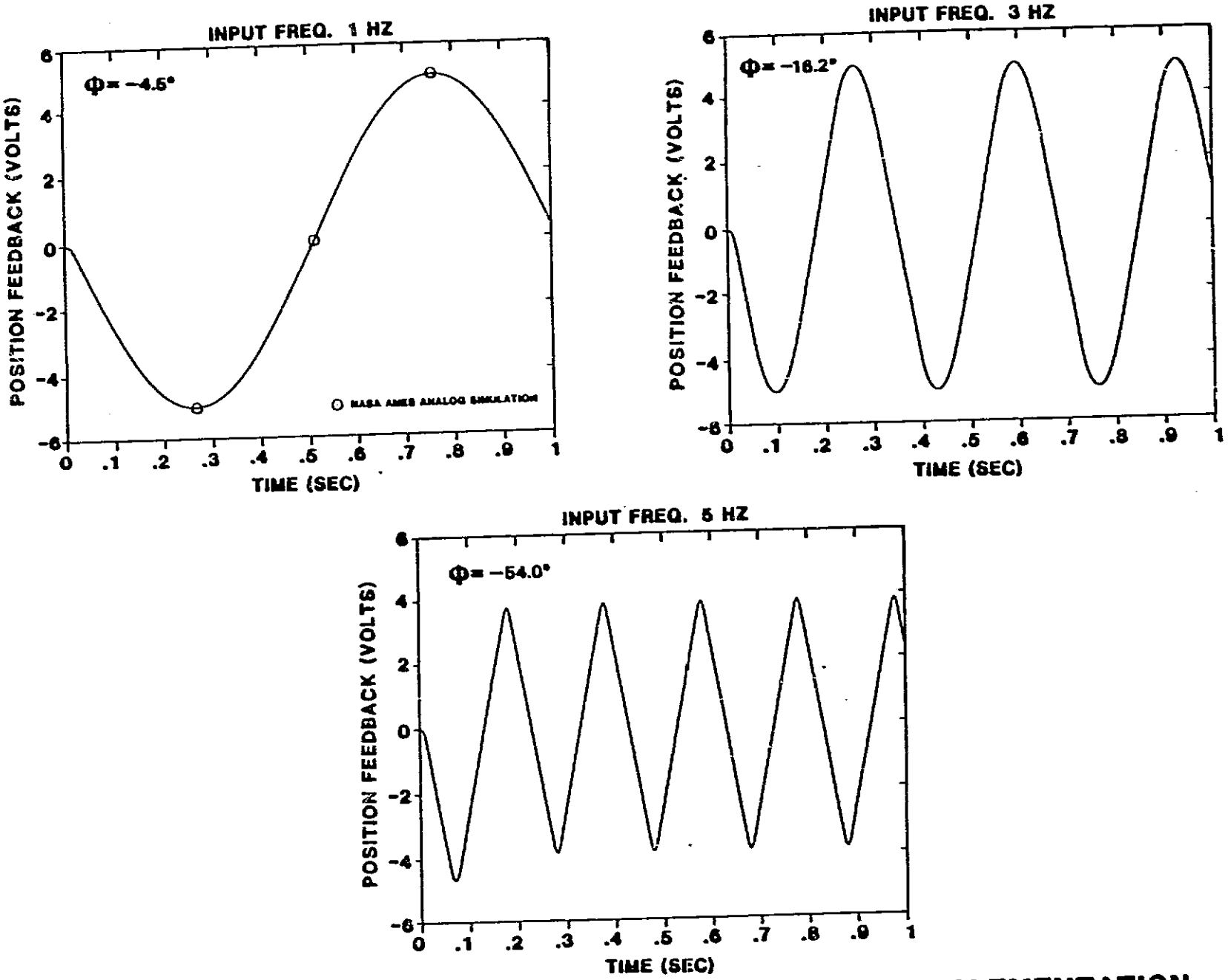
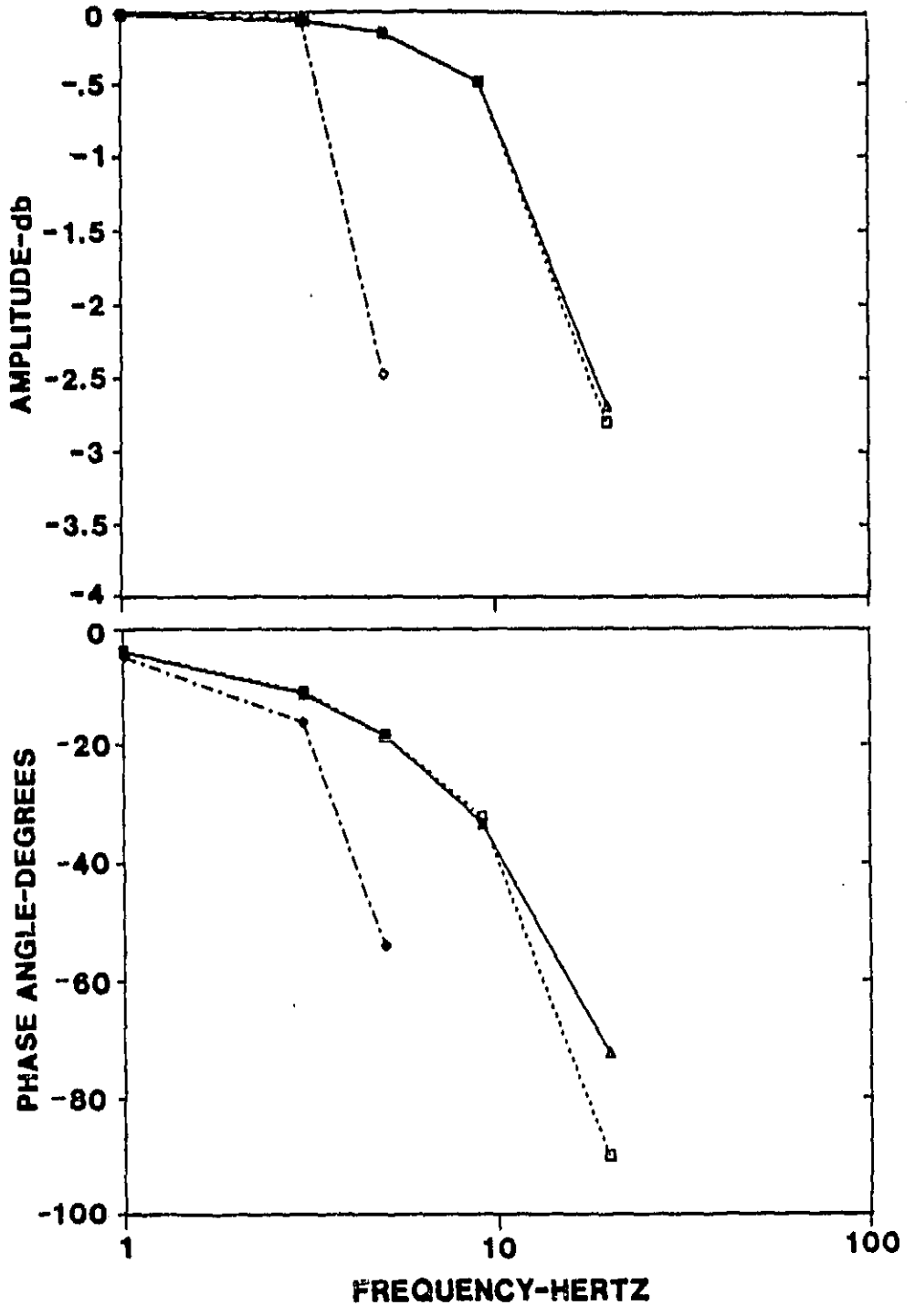


FIGURE 7 FREQUENCY RESPONSE, ANALOG INPUT & IMPLEMENTATION 50% FULL SCALE



——— LINEAR
– – – – 10% FULL SCALE (FLOW LIMITED)
— · — 50% " " " "

FIGURE 8 FREQUENCY RESPONSE, ANALOG INPUT & IMPLEMENTATION, EFFECTS OF FLOW LIMITER

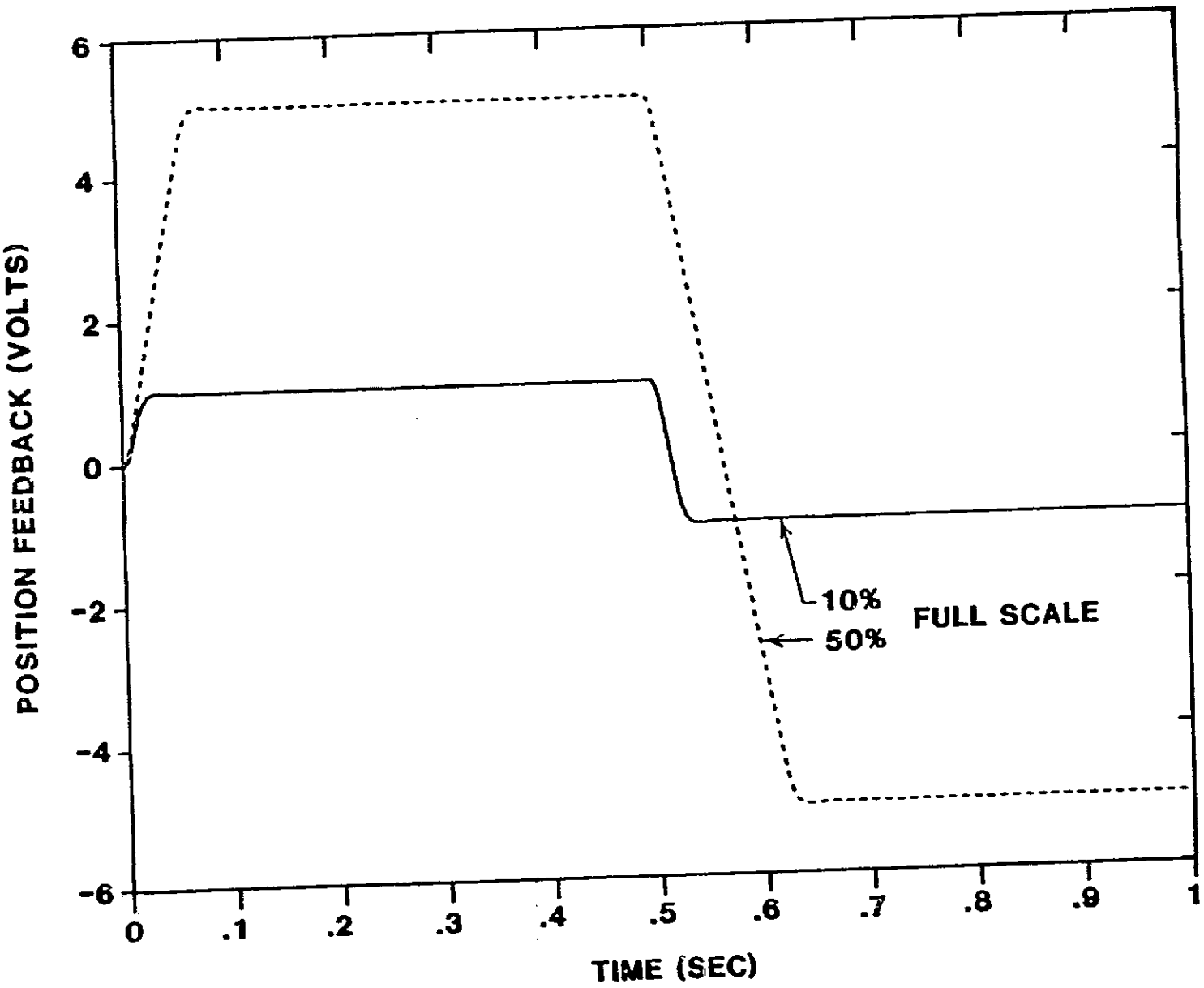


FIGURE 9 STEP RESPONSE, ANALOG INPUT & IMPLEMENTATION

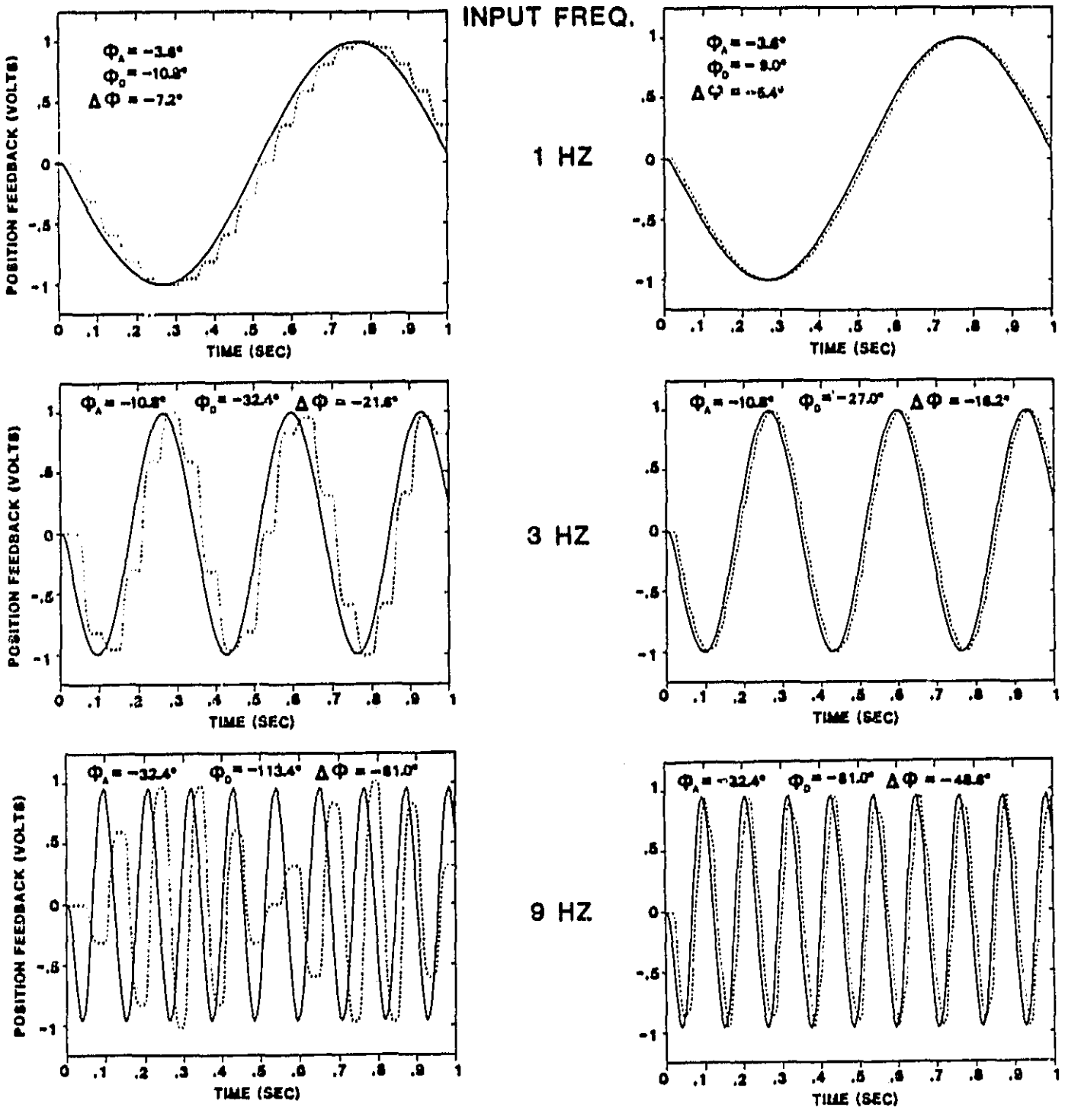


FIGURE 10 FREQUENCY RESPONSE, ANALOG IMPLEMENTATION, DIGITAL VS ANALOG INPUT, 10% FULL SCALE

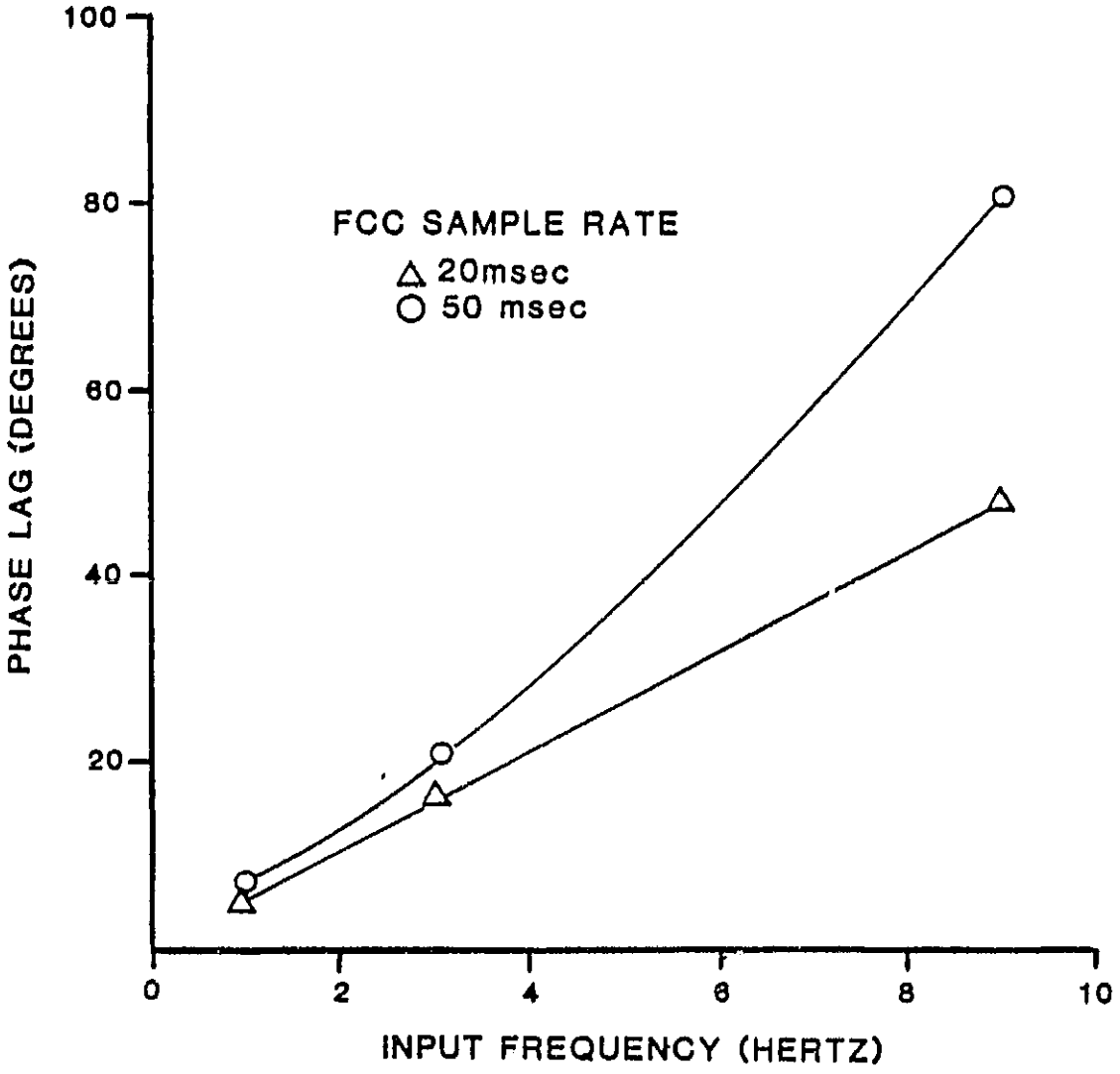


FIGURE 11 FREQUENCY RESPONSE (PHASE LAG) ANALOG IMPLEMENTATION, 10% FULL SCALE. EFFECTS OF DIGITAL INPUT

2. A digital FCC structure and an analog IRAS position loop control configuration.

3. A digital FCC structure and a digital IRAS position loop control configuration.

Analog FCC/Analog IRAS

This configuration is implemented to demonstrate the agreement with the AMES study simulation and to establish a benchmark against which the other configurations are evaluated. To analyze this configuration, all the integration steps and the interval of Sampler A of Figure 5-A are set to 1 msec, which was found adequate to duplicate analog behavior.

The dominant closed loop roots are related to the actuator dynamics. The damping ratio and the natural frequency are respectively $\zeta = .85$, $\omega = 26.2$ Hertz. The time responses to sinusoidal inputs are shown in Figure 6 for a 10% full scale (FS) input size and in Figure 7 for a 50% FS input size. These results, summarized in Figure 8, are consistent with those of the AMES simulation. The effects of the servovalve flow limiter, set at .561 cis, are clearly visible especially for high frequency inputs (5Hz) of high amplitude (50% FS). In this case, the flow limiter is responsible for a phase lag of about 50 degrees and an amplitude gain of -2.5 db. Throughout this report, the phase lag values represent the phase shifts between the input signals and the corresponding system outputs; negative values represent outputs lagging input signals. When a comparison is made with a "benchmark" configuration, then negative delta values represent additional output lag with respect to the outputs of that "benchmark" configuration.

The system response to step inputs is shown in Figure 9. Consistent with the previous cases, the effects of the flow limiter are more pronounced for large inputs (50% FS) than small inputs (10% FS).

Digital FCC/Analog IRAS

This configuration is implemented to analyze the effects of digitized FCC input commands. A major effect of digitized commands is to introduce phase lag in the system dynamic response as a result of (1) time skewness between the FCC and DCPU frame times, and (2) the sampling process which permits the system to sense and react to a continuously varying input at sampling times only.

In this analysis, it is assumed that the FCC and the DCPU are synchronized and that no time skewness exists between the corresponding frame times. If this were not the case, an additional lag would have to be added to the results of this analysis to account for that factor. The analysis is performed for an input amplitude equal to 10% FS to minimize the non-linear effects of the flow limiter. In Figure 10, the dynamic behavior of this configuration is directly

compared to the dynamic behavior of the previous configuration which was based on an analog FCC. The analysis considers 2 FCC sample rates: 20 msec and 50 msec; and, 3 input frequencies: 1 Hz, 3 Hz, and 9 Hz. The additional phase lag of this configuration, compared to the previous, is an increasing function of the input frequency and a decreasing function of the FCC sample rate (Figure 11). The divergent trend of the phase lag versus input frequency for a FCC sample rate of 50 msec might be due, at least in part, to the input frequency (9 Hz) approaching the Nyquist frequency (half of the sampling frequency, .5 x 20 Hz = 10 Hz). Step responses are shown in Figure 12, and in this case no appreciable differences exist between a digital and an analog FCC, and the two curves overlap almost exactly. It is interesting to note that the relatively high ratio of feedback loop gain to forward loop gain is responsible for the slight overshoot observable with both sinusoidal and step inputs. The word length does not appear to be a significant factor in this analysis. In fact, if the FCC and DCPU are assumed to be 16 bit processors then the system can measure 32767 discrete levels in either direction. If 12 bit Analog to Digital (A/D) and Digital to Analog (D/A) converters are used then 2048 discrete levels are available in the positive and negative direction. This corresponds to an accuracy of ± .05% or ± .005 Volts of ± 10 Volt FS. In the case of an 8 bit processor and conversion, an accuracy of ± 1.% or ± .1 Volt of ± 10 Volt FS is achievable, which is probably not adequate.

Digital FCC/Digital IRAS

This configuration is implemented to analyze the effects of digital position loop closure and the relationship between the FCC and DCPU frame times. In Figures 13 through 15, the dynamic behavior of this configuration is directly compared to the dynamic behavior of the previous configuration, based upon an analog position loop closure. The analysis is performed for an input amplitude equal to 10% FS to minimize the non-linear effects of the flow limiter and DCPU sample rates of 5 msec and 10 msec are considered. The following conclusions can be drawn:

1. No significant phase lag is added to the system dynamics by digitizing the position loop closure.

2. The overshoot characteristics, also observed in the previous configuration, are much more pronounced in this configuration. In the case of a sinusoidal input, the overshoot is a decreasing function of the DCPU and FCC sample rates. For a 50 msec FCC sample rate, the overshoot is 10.8% in the case of a 5 msec DCPU frame time and 25.5% for a 10 msec DCPU frame time (Figure 13). For a 20 msec FCC sample rate, the corresponding values are .1% and 7.3% respectively (Figure 14). The % overshoot does not change with the frequency of the input as long as the operating point is within the linear range of the servovalve. In the case of a step response (Figure 15), the overshoot characteristics improve with faster DCPU frame times but do not change as a function of FCC frame time.

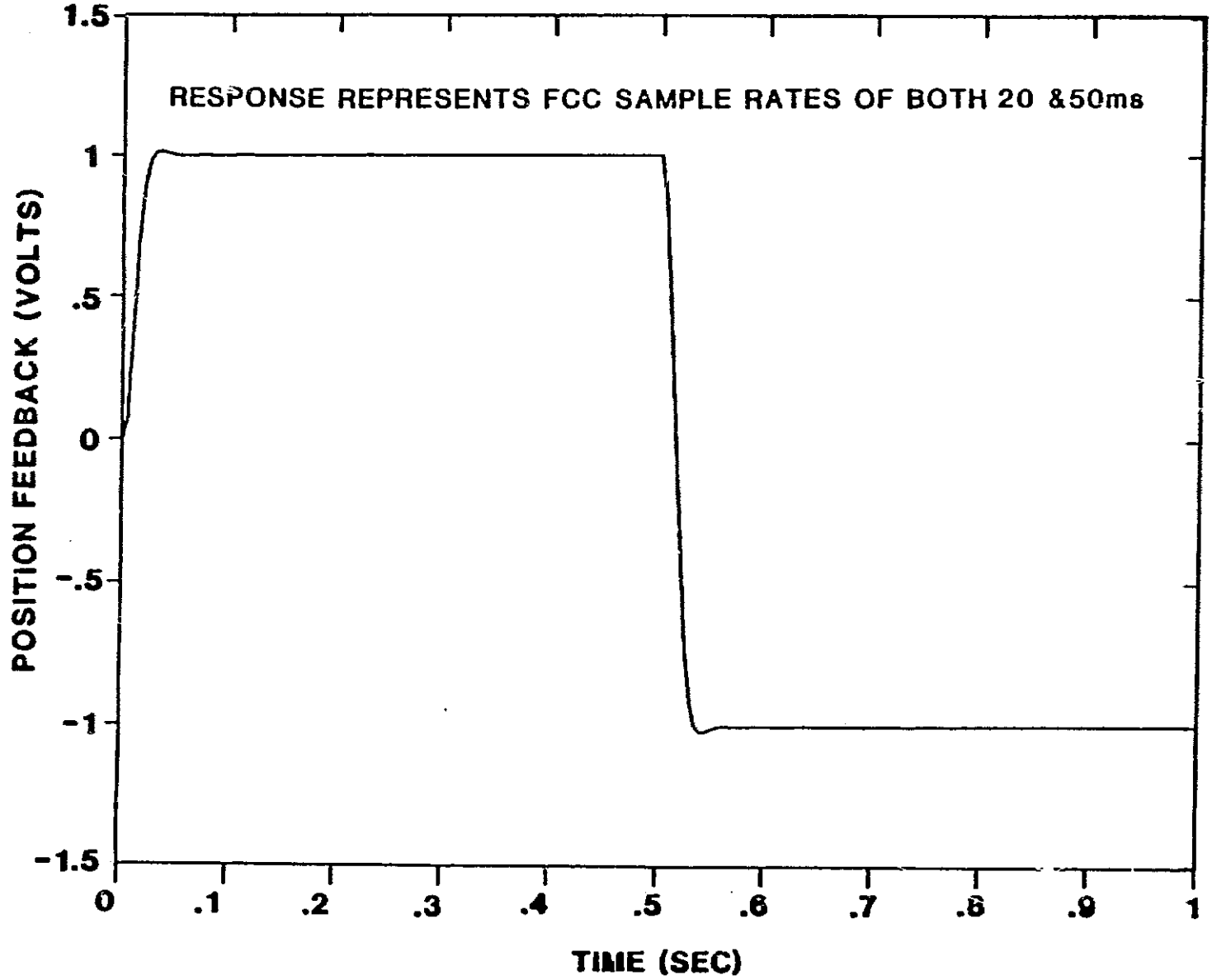
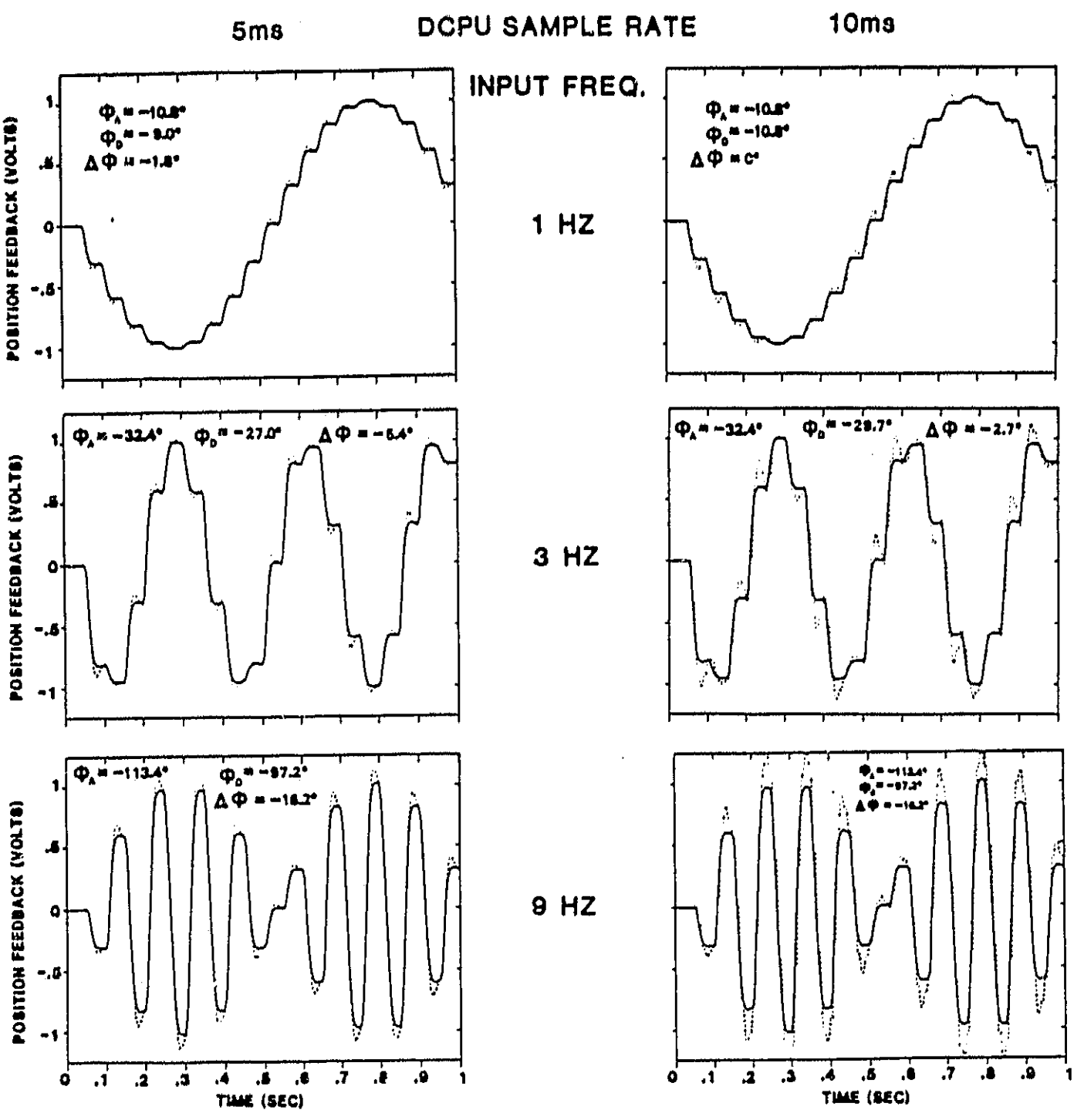


FIGURE 12 STEP RESPONSE, ANALOG IMPLEMENTATION, DIGITAL VS ANALOG INPUT, 10% FULL SCALE



——ANALOG LOOP CLOSURE Φ_A

----DIGITAL LOOP CLOSURE Φ_D

FIGURE 13

FREQUENCY RESPONSE, 10% FULL SCALE, FCC SAMPLE RATE 50ms
DIGITAL VS ANALOG POSITION LOOP CLOSURE

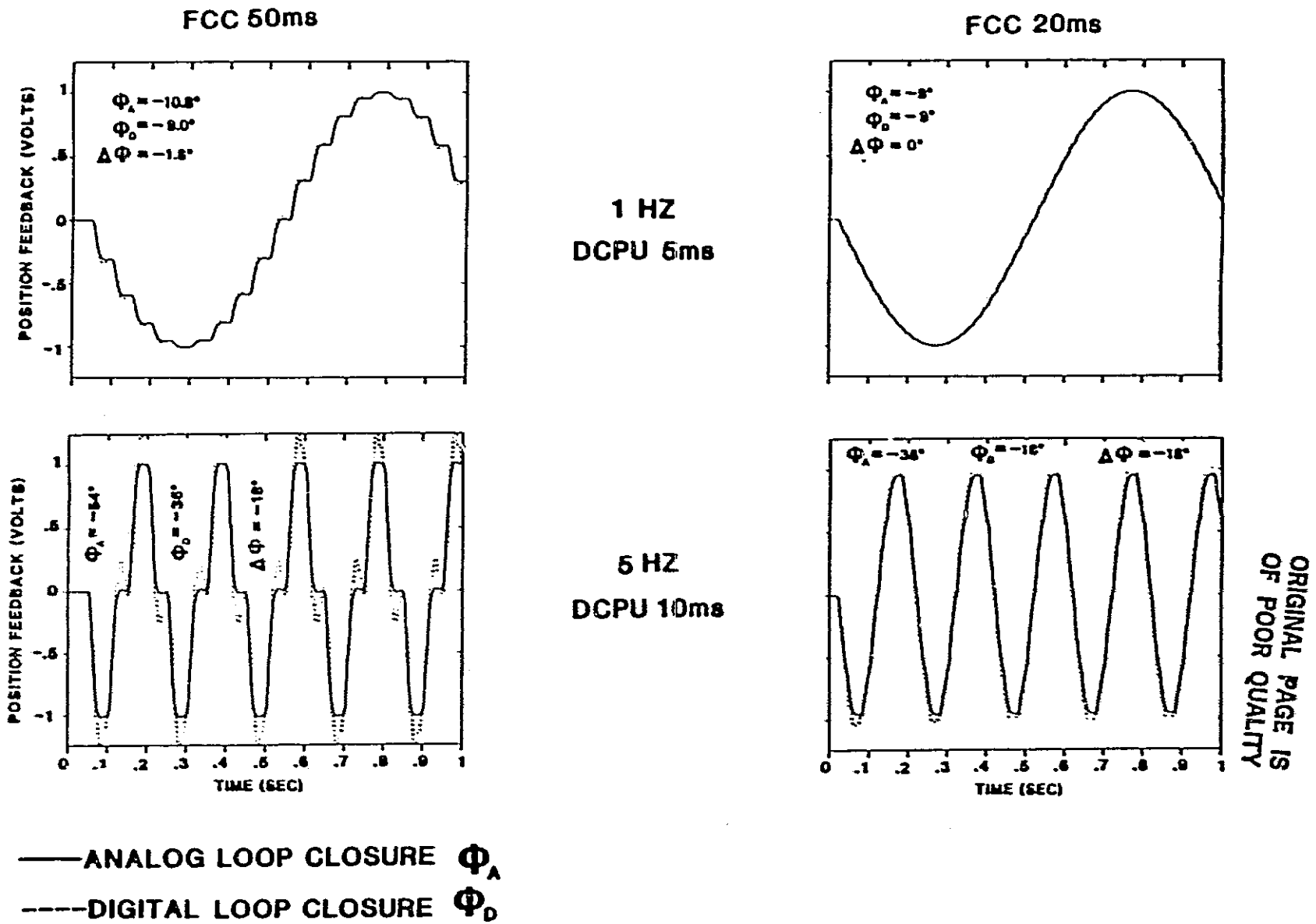
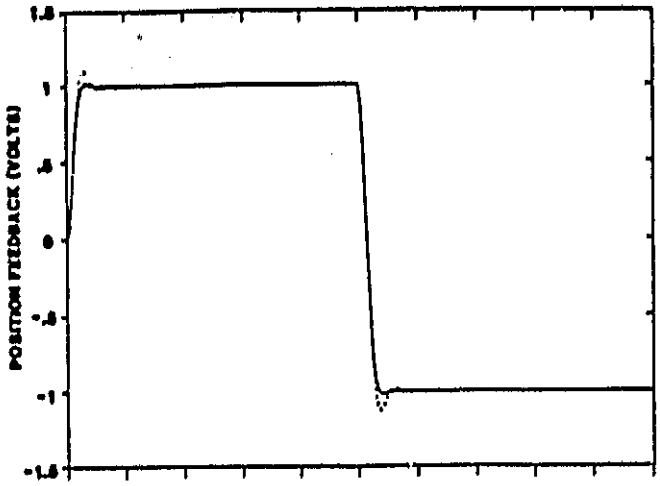
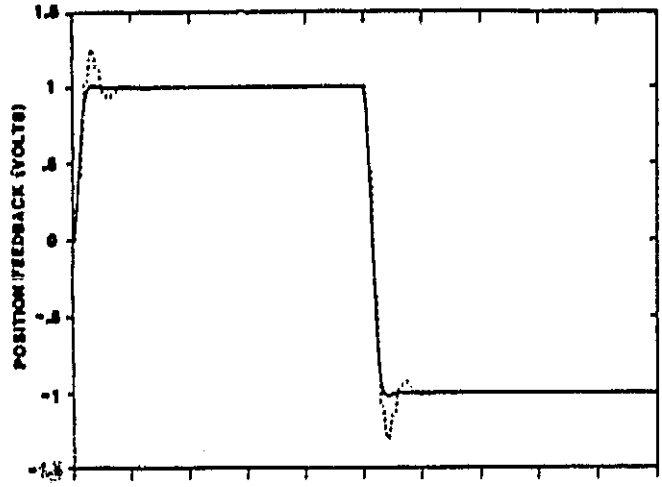


FIGURE 14 FREQUENCY RESPONSE, 10% FULL SCALE, DIGITAL VS ANALOG POSITION LOOP CLOSURE

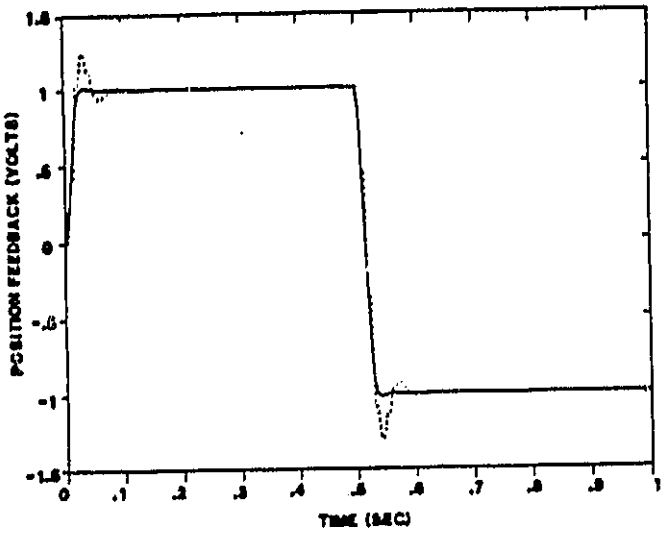
FCC 50ms
DCPU 5ms



FCC 50ms
DCPU 10ms



FCC 20ms
DCPU 10ms



——ANALOG LOOP CLOSURE

----DIGITAL LOOP CLOSURE

FIGURE 15 STEP RESPONSE, 10% FULL SCALE, DIGITAL VS ANALOG POSITION LOOP CLOSURE

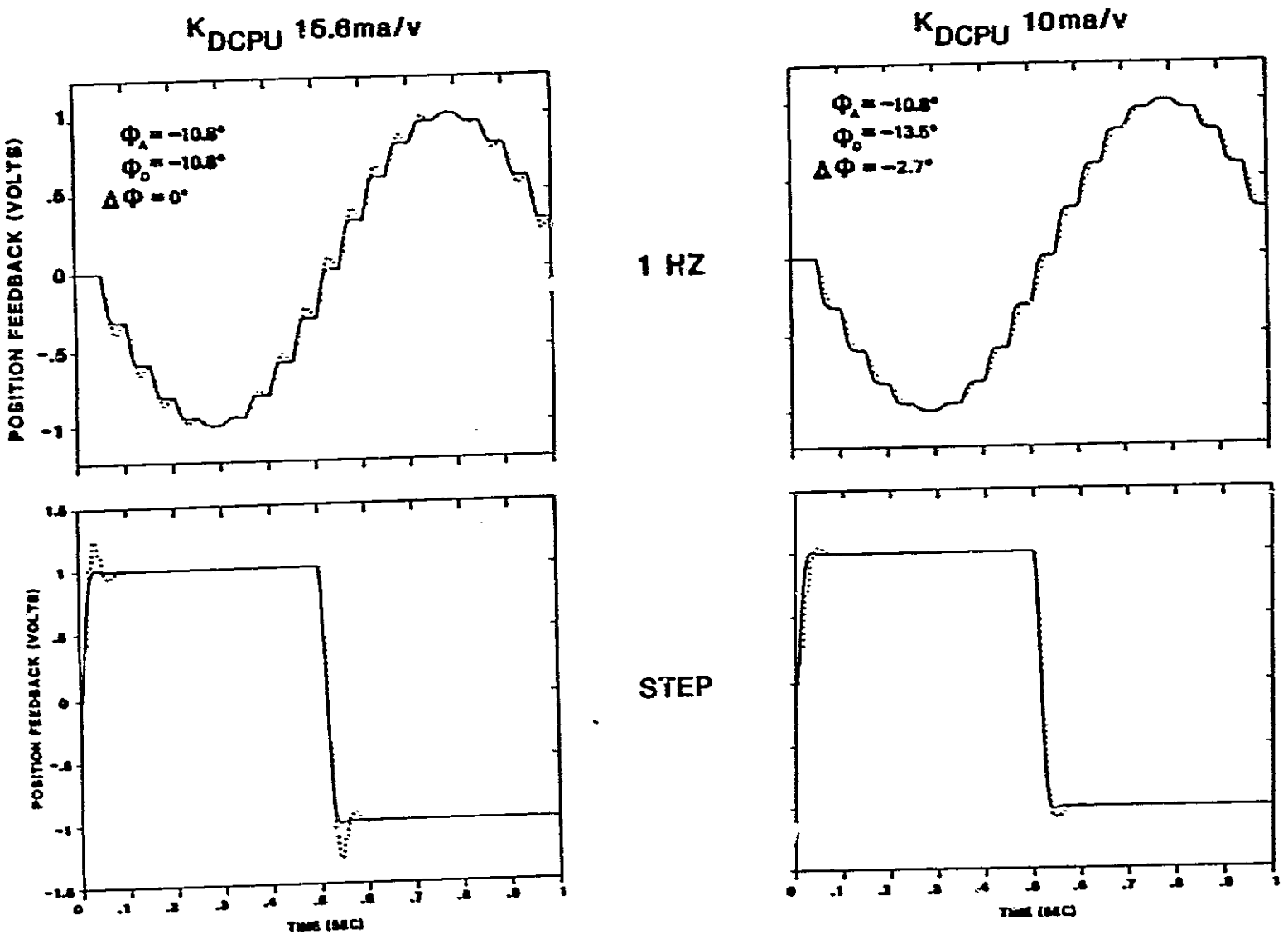


FIGURE 16

FREQUENCY/STEP RESPONSE, 10% FULL SCALE, FCC SAMPLE RATE 50ms, DCPU 10ms, EFFECTS OF FORWARD PATH GAIN

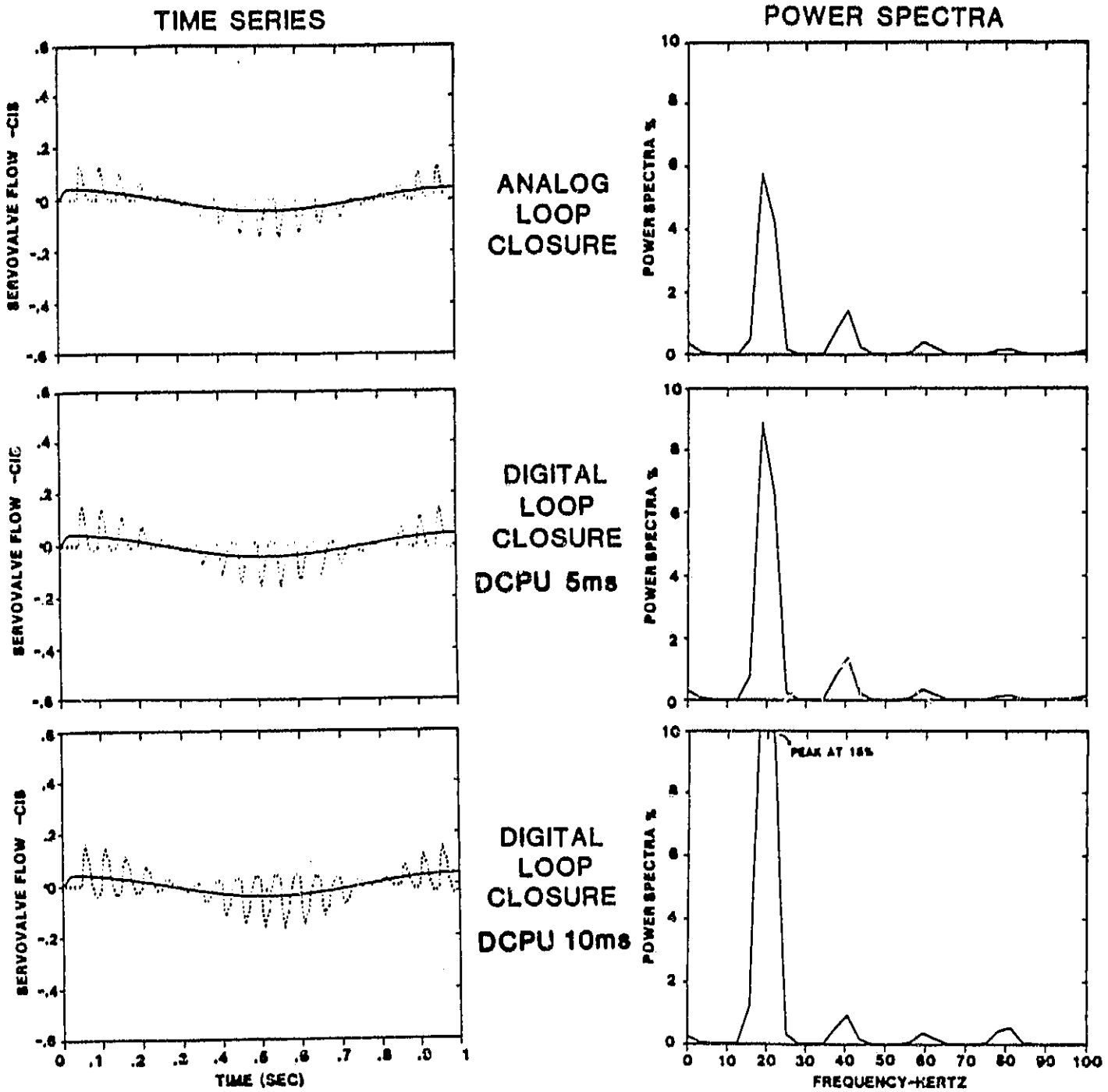
The overshoot characteristics can be reduced substantially by reducing K_{DCPU}, the forward path gain (Figure 16). In the case of a 1 Hz input sine wave, the overshoot was reduced from 25.5% to .2% by reducing K_{DCPU} from 15.6 ma/V to 10 ma/V. The corresponding overshoot values in case of a step input are 25.5% and 5% respectively. A side effect of the K_{DCPU} reduction is an additional phase lag of 3 degrees for a 1 Hz input.

Effects of Digitized Input Commands on Servovalve Activity

In any configuration which uses digital FCCs, the input to the servoactuator controller is in a digitized form, irrespective of the servocontroller architecture. When a continuous input, such as a sine wave or ramp, is digitized, it is converted into a series of small step inputs. The results of the analysis previously described confirms that the high frequency and the small size of these steps are such that the dynamic response of the actuator is generally only slightly effected by the digitization process. However, the effect of the digitization process on the servovalve activity is quite dramatic, and may cause undue wear and fatigue not only of the servovalve spool but also of the hydraulic lines. This in turn might produce premature failure or might negatively effect the preventative maintenance schedule and the projected life of the equipment.

In Figure 17, a comparison is made between the servovalve flow in an all analog system and the corresponding flow in three configurations which use digital FCCs. A FCC with a frame time of 50 msec and 1 Hz sine wave input of amplitude equal to 10% FS were considered for this analysis. In all three digital input cases, the servovalve activity is clearly much higher than in the case of an analog input. It is also evident that the activity is higher when digital, rather than analog position loop closures are considered. In the case of digital loop closures, the servovalve activity increases with the DCPU frame time. In all cases, the power spectra density has a very high peak at 20 Hz, which is the input sample rate. Minor peaks also occur at integer multiples of 20 Hz; the amplitude of these peaks is a decreasing function of the frequency.

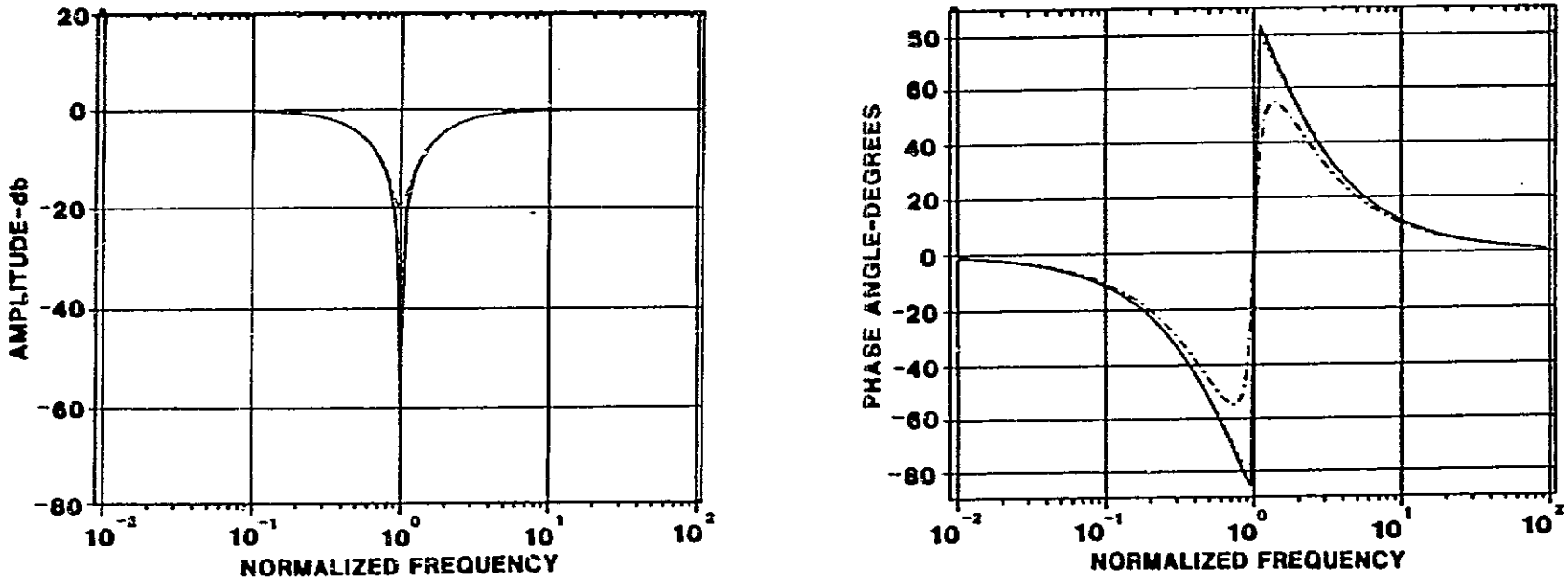
A possible solution to minimize the servovalve activity in the case of a digital loop closure, is to sample command and feedback signals at the same time. By doing this, the servovalve driver, the difference between command and feedback, is also computed at the sample time only and does not vary between sample times as the analog feedback does. For a ramp input, and in a condition of dynamic equilibrium, a constant error is developed between sampled input command and sampled feedback signal, and the resultant servovalve response is a constant spool displacement to develop a constant flow rate. The disadvantage of this solution is that it effectively requires the closure of the position control loop at a rate equal to the FCC frame time, and this relatively low computational rate will degrade the actuator dynamic performance to unacceptable levels.



—— ANALOG INPUT/ANALOG IMPLEMENTATION

----DIGITAL INPUT/ANALOG OR DIGITAL LOOP CLOSURE

FIGURE 17 EFFECTS OF DIGITAL INPUT ON SERVOVALVE ACTIVITY, 10% FULL SCALE, FCC SAMPLE RATE 50ms, INPUT FREQUENCY 1 HZ



DAMPING

—— .001

– – – – .010

— · — .100

FIGURE 18 NOTCH FILTER CHARACTERISTICS

A more realistic solution to the problem, applicable to both digital and analog closures, is to apply notch filters on the digital input command. Notch filters can drastically reduce the frequency content of input functions at a given frequency while having little impact on all other frequencies. In Figure 18, the characteristics of a generic notch filter are shown. To satisfy our requirements, the corner frequency of the filter is equal to the FCC frame time and the natural damping is selected to be .1, a compromise between the requirements of high suppression at the corner, and low distortion in the remaining frequency region. In Figures 19, 20, and 21, the effect of applying a notch filter to a digitized input is shown for an analog position loop closure and for digital position loop closures with DCPU frame times of 5 and 10 msec. The results are summarized in Figure 22. It is evident that the notch filter drastically damps the sampling frequency in all cases. As an example, in the case of a 1 Hz sine wave input of amplitude equal to 10% FS and a DCPU frame time of 5 msec, the 20 Hz power spectrum peak is reduced from a value of 9.0% to a value .2% by the notch filter. However, it is interesting to note that the second harmonic (at 40 Hz) is reduced by a much smaller amount; in this case, the power spectra value is reduced from 1.6% to .7%.

The effects of the notch filter on the actuator dynamics are shown in Figure 23 for a 1 Hz sinusoidal and a step input. Only small variations exist between the analog loop closure and the digital loop closure with a 5 msec frame time. Additional distortion appears in the case of a 10 msec frame time. The phase lag induced by the notch filter is about 9 degrees, for a sinusoidal input and DCPU frame time equal to 5 msec. In the case of a step input and digital loop closure, the time to reach 90% of the command input is more than doubled when a notch filter is applied to the input (Figure 24). The results of this analysis are preliminary in nature, however, the following conclusions can be drawn:

1. A notch filter, applied to the digitized input command from the FCC, can effectively damp the servovalve activity at the sample frequency. The effect on higher harmonic is not significant.

2. A notch filter adds a phase lag to the overall system response which increases as a function of the DCPU frame time.

Issues which need to be further investigated include:

1. Analysis of the effect of high frequency servovalve activity on the maintenance schedule and life expectancy of the actuation system.

2. Effects of notch filter induced phase lags on overall system performance.

3. Methods, other than input filtering, to reduce the effects of digitized inputs on servovalve activity.

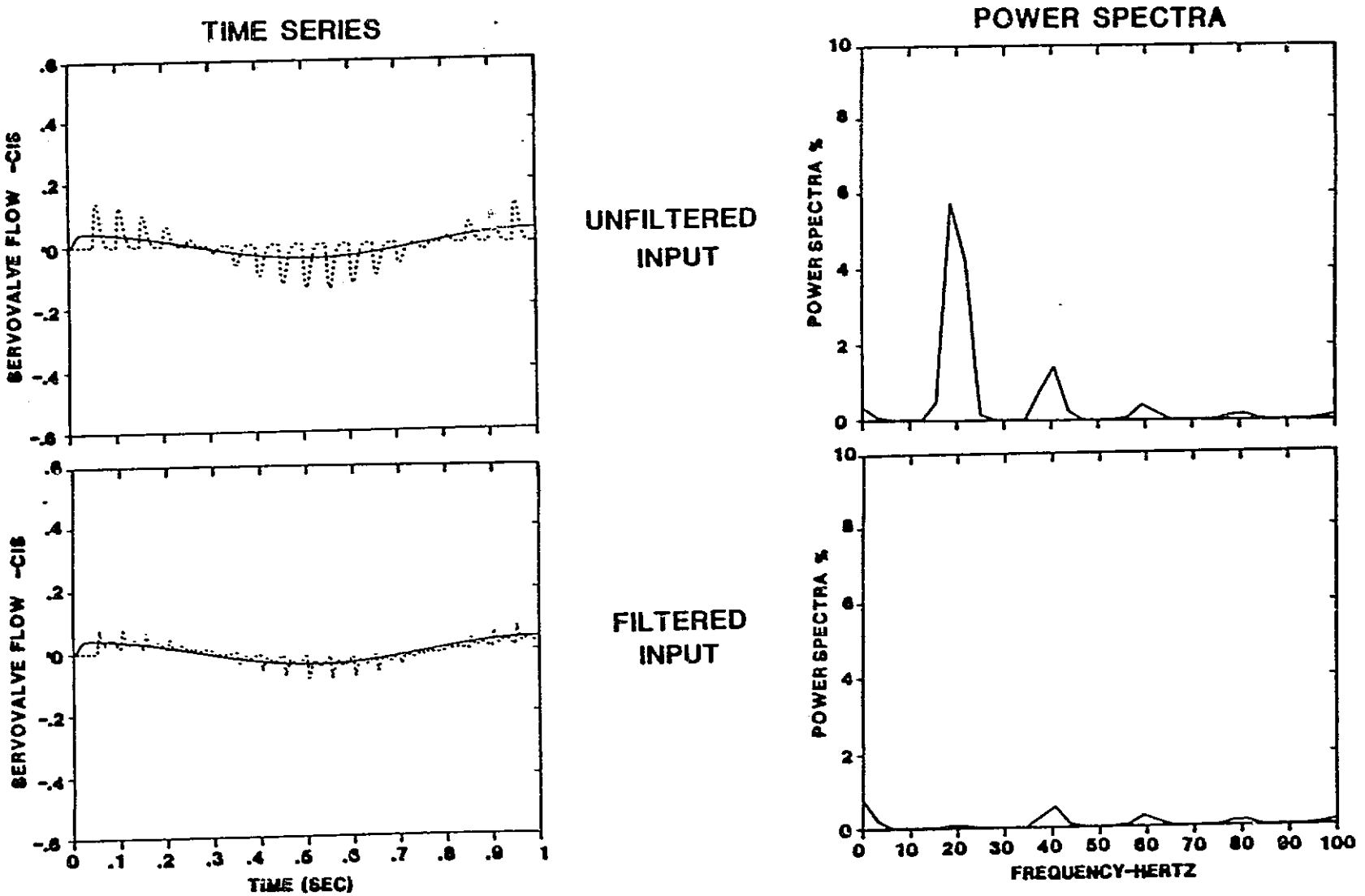


FIGURE 19
NOTCH FILTER COMPENSATION OF DIGITAL INPUT/ANALOG IMPLEMENTATION
10% FULL SCALE, FCC SAMPLE RATE 50ms, INPUT FREQUENCY 1 HZ

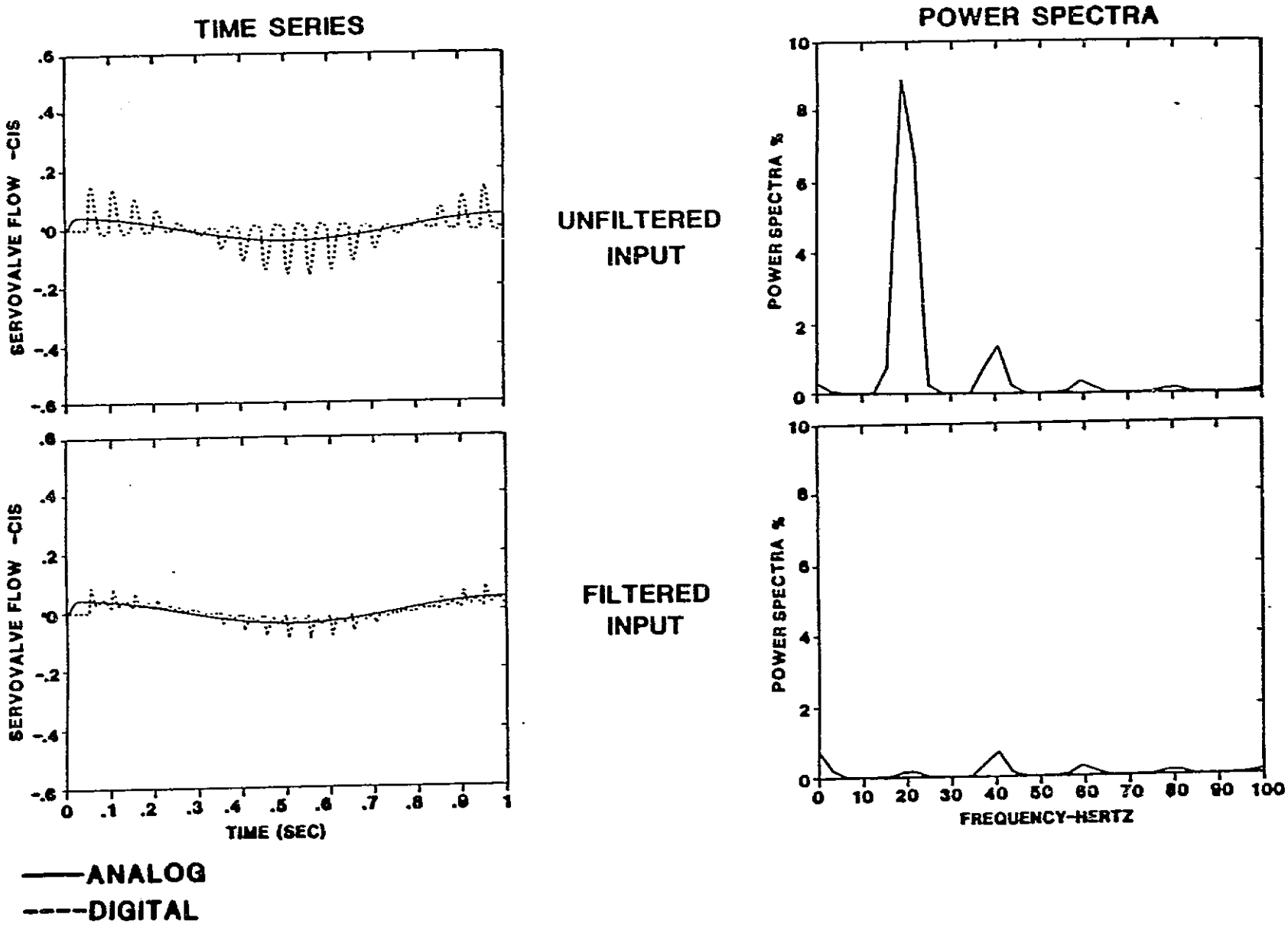


FIGURE 20

NOTCH FILTER COMPENSATION OF DIGITAL INPUT/DIGITAL IMPLEMENTATION 10% FULL SCALE, INPUT FREQUENCY 1 HZ, FCC SAMPLE RATE 50ms, DCPU 5ms

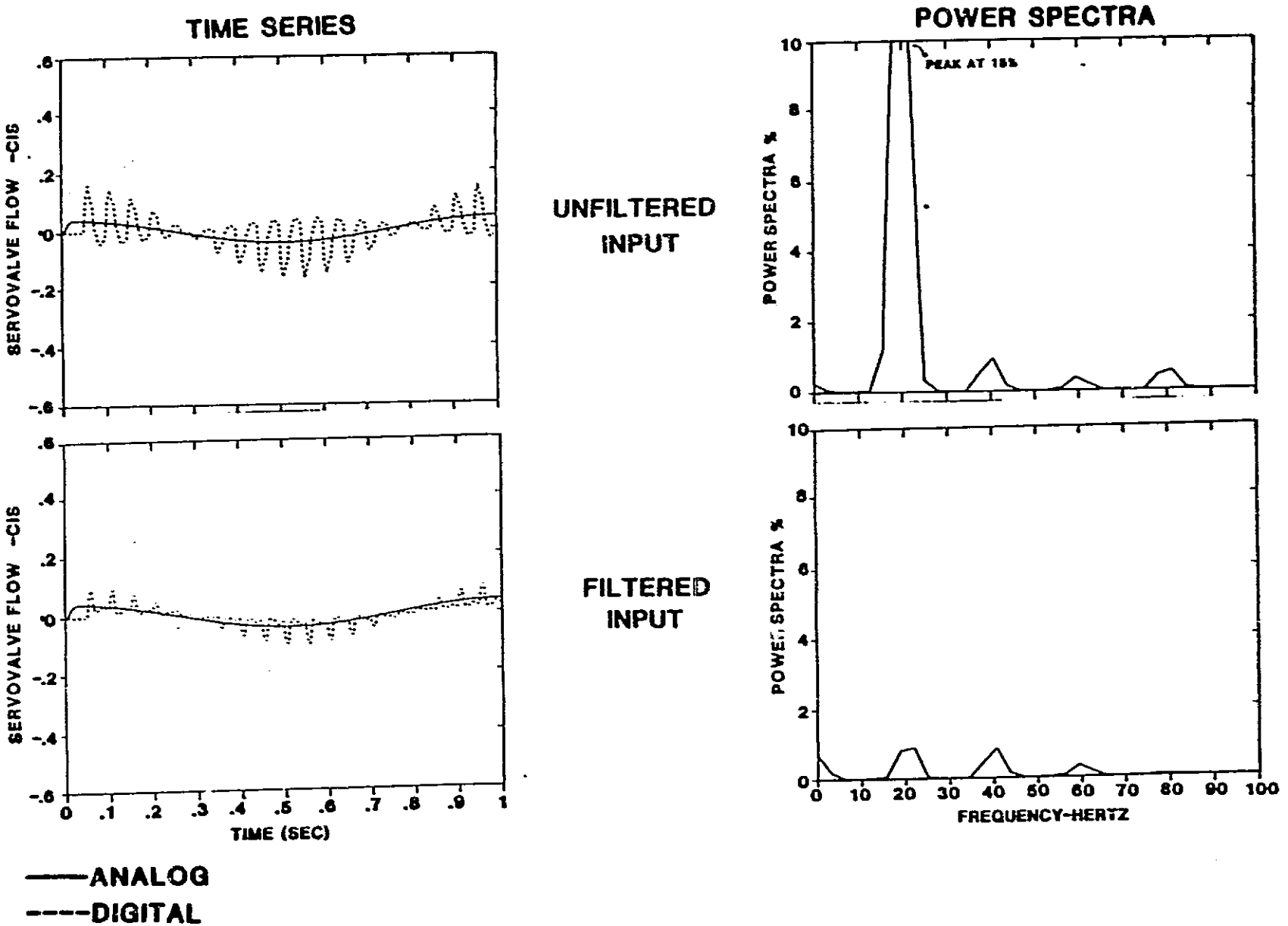
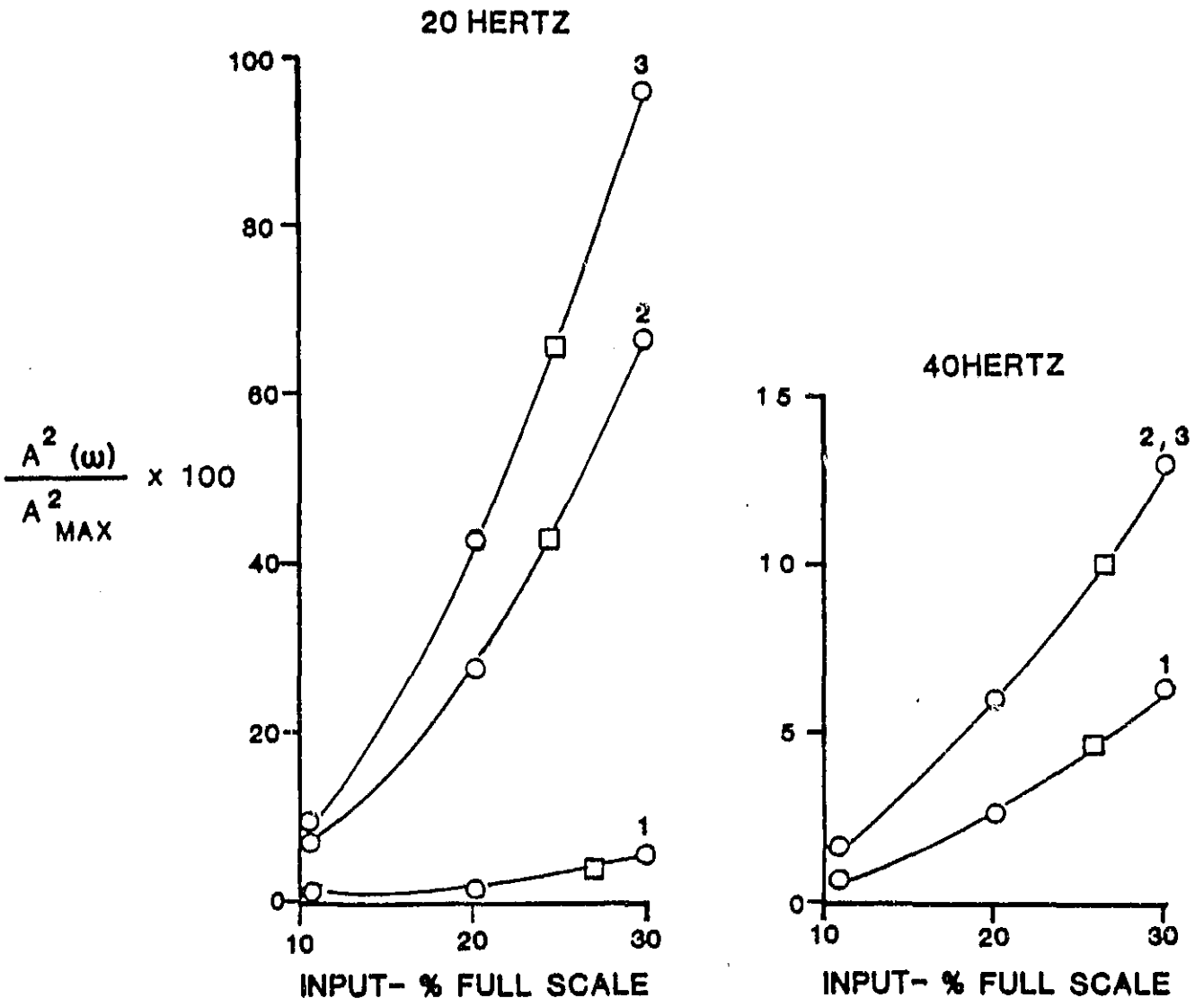


FIGURE 21

NOTCH FILTER COMPENSATION OF DIGITAL INPUT/DIGITAL IMPLEMENTATION
10% FULL SCALE, INPUT FREQUENCY 1 HZ, FCC SAMPLE RATE 50ms, DCPU 10ms



1 - DIGITAL IMPLEMENTATION (DCPU 5msec) -FILTERED INPUT

2 - ANALOG IMPLEMENTATION- UNFILTERED INPUT

3 -DIGITAL IMPLEMENTATION (DCPU 5msec) - UNFILTERED INPUT

O- 1 HERTZ INPUT

□- RAMP INPUT (100%F.S./SEC)

$A_{MAX} = .561$ CIS

FIGURE 22

EFFECTS OF DIGITAL INPUT ON SERVOVALVE ACTIVITY, POWER SPECTRA AMPLITUDE OF FLOW RATE AT 20 &40 HERTZ

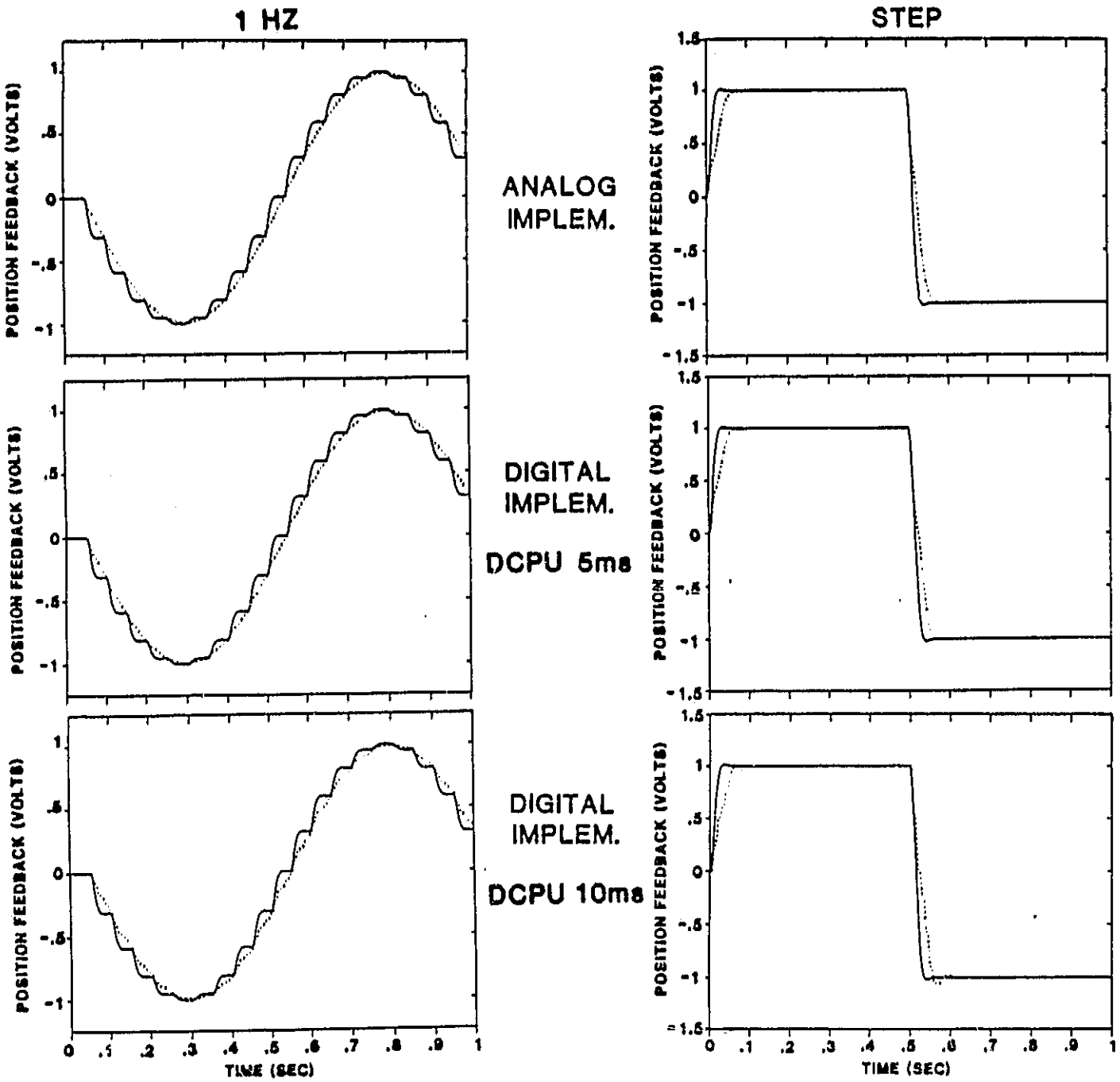


FIGURE 23
EFFECT OF NOTCH FILTER ON POSITION CONTROL, 10% FULL SCALE

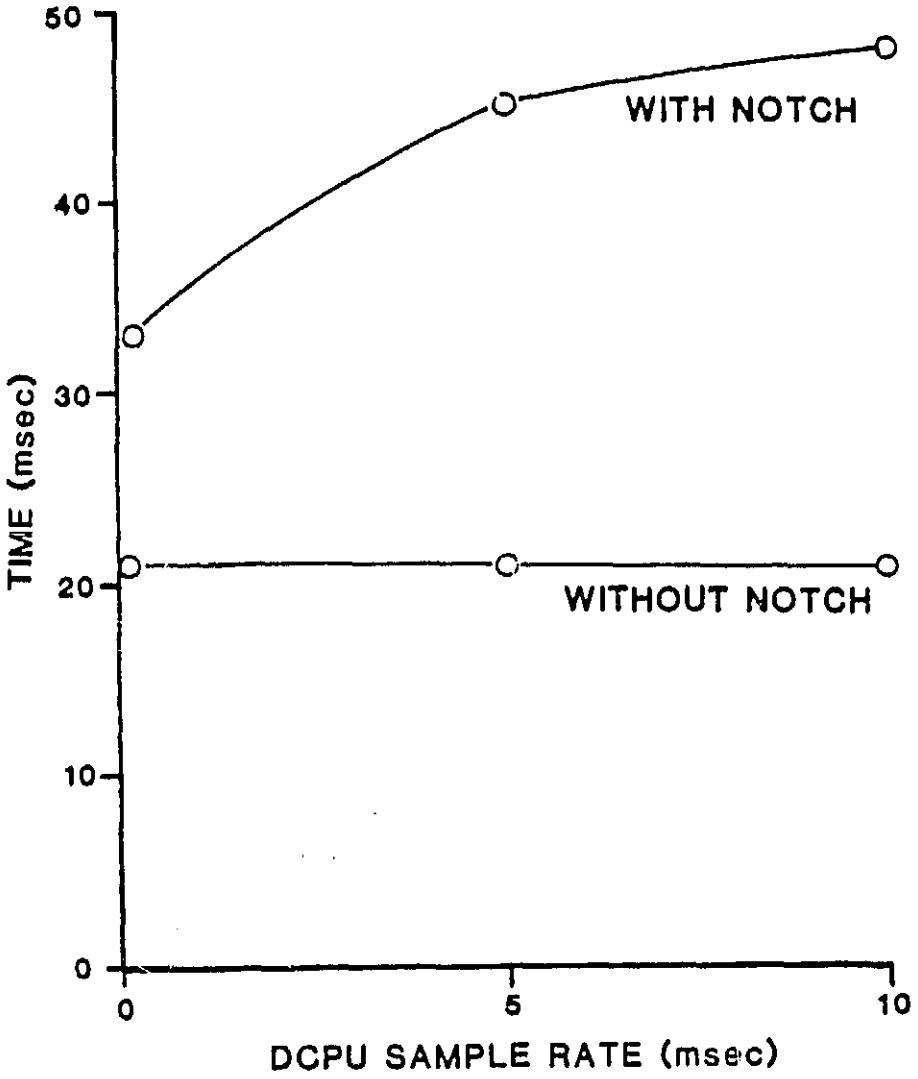


FIGURE 24 EFFECT OF NOTCH FILTER ON TIME TO REACH 90% OF COMMAND, 10% FULL SCALE

HARDWARE REQUIREMENTS

The objective of this effort is to provide a flexible, powerful environment where the feasibility of an Intelligent Redundant Actuation System can be demonstrated and where related issues can be analyzed and competing configurations evaluated. Furthermore, this objective must be achieved in the most cost effective manner. This effort is not involved in resolving or addressing packaging and environmental issues and compliances with reliability and safety requirements typical of flight worthy implementations.

The high level program requirements translate into the two following hardware requirements:

1. Use of off-the-shelf commercially available products must be maximized. Examples of this type of equipment are the DCPUs and the Arbitrator's microprocessors.

2. Use of the existing TAFCOS hardware or hardware design must be maximized without compromising the IRAS research objectives. Examples of this type of equipment are all the interfaces with the actuator.

These requirements are applied to all the major hardware elements of the IRAS: DCPU; Arbitrator; CIU; Tests; and Interfaces.

DIGITAL COMMAND PROCESSING UNIT (DCPU)

The major decision relative to the DCPU is the selection of an appropriate microprocessor which can meet all the current and projected computational requirements and which is well supported by a vast assortment of software and hardware tools. For this purpose, a set of evaluation criteria have been established and the selection has been performed consistent with those criteria. The two major high level evaluation criteria for the microprocessor are: the computational capacity and the availability of high quality development tools.

The computational capacity of the selected microprocessor system determines the maximum achievable algorithmic complexity and iteration rate. Processor attributes which effect the computational capacity are the architecture, the clock frequency, the characteristics of the compiler when a high order language (HOL) is used.

A processor architecture consists of the instruction set, word size, address space, registers set and addressing modes. These components combine to determine the minimum number of instructions required to implement specific algorithms. Architectural attributes favoring high computational capacity are a large word size (16 or 32 bits/word rather than 8 bits/word); a powerful instruction set; a large address space; a register set

which includes several 16 or 32 bit general purpose registers; and several addressing modes which allow great flexibility in accessing data. The instruction execution time is determined by the CPU clock speed, by the number of clock cycles required to execute an instruction, and by the memory access time. The architecture and the instruction execution time, ultimately determine the computational capability of the system.

Software and hardware development tools must also be evaluated. Software development tools are programs such as HOL compilers, HOL symbolic debuggers, assemblers, linkers, coverage analyzers, etc. Hardware development tools are processor specific electronic devices such as logic analyzers and in-circuit emulators, which aid the development of system hardware and software. A HOL compiler and HOL symbolic debugger are the two priority software tools required for the IRAS project. A logic analyzer is a very desirable hardware tool.

The HOL compiler is notably the most critical software tool and the quality of the code generated is the primary attribute. Code quality is determined by the number of machine level instructions generated for each high level language statement. A high quality compiler will usually generate two or three instructions for each HOL statement, if the target architecture is powerful. A good compiler will not generate unnecessary assembly code, which unduly increase the execution time. It must be very reliable and produce clear diagnostic messages. Also very important is a HOL symbolic debugger which allows the user to control program execution and examine and modify the program at the HOL level.

The most important hardware tool is a logic analyzer which is used to trace program execution in real-time, and to detect timing and synchronization errors. A logic analyzer is also useful in monitoring the execution time of critical program sections.

An evaluation has been made of commercially available microprocessors for performing the function of the DCPU in the IRAS system. The evaluation activity measured each microprocessor against the DCPU requirements and reached the conclusion that one microprocessor is superior to all others for this effort. The following describes the results of the evaluation and the justification of the selection.

The experimental nature of the IRAS system requires a high level of computational capacity. This requirement limited the in-depth analysis to the high performance microprocessors, specifically the Intel 8086 family, the Motorola 68000 family, the National 16000/32000 family, and the Zilog Z8000 family.

Early consideration was also given to the Intel 8051, 8048 and 8096 processors. This series of processors are considered suitable for flight worthy implementations because they contain several on-chip support functions which reduce the number of components required to implement a system, and ultimately reflect in reduced cost and increased reliability. However, their weak support and low computational capacity make them inappropriate for the IRAS lab environment.

In the early stages of the evaluation, the National Semiconductor 16000/32000 processor family was determined to be unacceptable. While this family of processors has a very powerful architecture, it lacks hardware and software support. The same lack of hardware and software support was also determined for the Zilog Z8000 family.

Therefore, the analysis was focused on the Intel 8086 and the Motorola 68000 families. Both processors were found to be strongly supported in the hardware and software. Hardware support includes several board level products, logic analyzers and in-circuit emulators. Software support includes several resident and cross HOL compilers, operating systems, and HOL symbolic debuggers. Hardware and software support, instruction set, and addressing mode, are equivalent for both processors. However, evaluation of the computational capacities of the 8086 and 68000 revealed that the 68000 is a more powerful machine than the Intel 8086. Architectural advantages of the 68000 are the register set, the partial support of 32 bit integers, and a 20% faster clock rate, so that the average execution time of several programs was 30% faster in the 68000 than in the Intel 8086.

Some of the major architecture differences between the two machines are:

- The 68000 has sixteen 32 bit registers which are used in a consistent manner by the instruction set; while the 8086 has twelve 16 bit registers that are not used consistently by its instruction set.

- The 68000 has more available registers with more bits per register than the 8086. This reduces the number of instructions to load and store registers, thus reducing the number of instructions required to implement an algorithm.

- The consistency in which the 68000 uses its register set also minimizes the number of instructions required to implement an algorithm. In fact, this eliminates the need for "register transfer" instructions that would be required if the register containing the instruction data could not be used by the instruction.

The architectural advantages of the 68000 are proven by various benchmarks where the 68000 outperforms the 8086 by 20 to 40 percent relative to execution time. Thus, the high level of

hardware and software support and the superior computational capability of the 68000 processor family is the justification for its selection for the IRAS project.

ARBITRATOR

The primary function of the Arbitrator is to determine the failed or operational status of each channel and set appropriate status flags to the correct value based on information from all channels. The four health status flags, one for each channel of the quad system, are:

1. Transmitted back to the DCPUs so that appropriate gain compensation can be made by all operational channels.

2. Transmitted to the CIUs so that the failed channels can be appropriately isolated.

The function of the Arbitrators is very critical, and therefore, in a flight critical environment the Arbitrator should be highly reliable and fault-tolerant. In the laboratory environment of the IRAS, issues related to flight worthy implementations are not addressed; instead emphasis is placed on capabilities to emulate and analyze a variety of functional implementations so that the relative merits can be established. Under these circumstances it is best to implement the logic of the Arbitrator in software programs hosted in a dedicated microprocessor. In this case, the logic can be easily modified so that different concepts can be analyzed; the effects of computational and data transfer delays can be realistically assessed; and redundant configurations for the Arbitrator can be emulated and evaluated.

A secondary function of the Arbitrator is to support the tasks of simulation mode control and fault insertion; for this purpose directives are sent from the TCT to the Arbitrator and then routed to the DCPUs via common backplane connectors, and to the CIUs via dedicated digital and analog links. A dedicated Motorola 68000 has been selected for the Arbitrator. Clearly the power of the 68000 exceeds all the current and projected needs of any conceivable Arbitrator configurations. In addition, there are many significant justifications for using the same microprocessor used for the DCPU such as:

1. The use of a common language, development environment and supporting tools.

2. The commonality and flexibility of the interfaces including those achievable through backplane connections.

3. The possibility of using the excess power for future research tasks and for auxiliary tasks such as simulation mode control and fault insertion.

4. Cost of savings resulting from buying a number of industrial single board processors.

CONTROL INTERFACE UNIT (CIU)

The primary function of the CIU is to perform all the data conversion necessary to interface directly with the actuator, including the coil current drivers, the LVDT modulation and demodulation, and the actuator solenoid. An optional function of the CIU is to perform the outer loop, or position loop, closure. An alternate approach is to close that loop within the DCPU. The implication of the choice between these two methods are significant not only relative to the overall system dynamic behavior, but also to the internal structure of the CIU and to the DCPU/CIU interfaces. Some important implications are discussed here.

If the loop closure is performed within the DCPU, then the CIU is only required to convert and to scale the input command received from the DCPU using a constant scale factor. In fact, the tasks of computing the position errors and of adjusting the control loop gain as a function of the failure modes or other factors, is performed internally by the DCPU. The structure of the CIU current driver is rather simple in this case and it can be implemented by using fixed gain operational amplifiers. The only needed data exchange to support these tasks is the transfer from the DCPU to the CIU of one variable appropriately scaled: the position error. A simple analog interface which uses standard D/A converters can be used for this purpose.

If the position loop closure is performed within the CIU, then the internal structure of the CIU and the DCPU/CIU interface becomes more complex. In fact, the variable loop gain is still computed by the DCPU as a function of the failure mode, airplane state and control mode. However, it must be actually implemented in hardware, within the CIU and then applied within the CIU, to either the position command (from the DCPU) and the LVDT position feedback or, preferably, directly to the position error. In either case, the resultant structure of the CIU current driver is rather complex and requires the use of variable gain operational amplifiers. The gains must be selectable within a discrete but rather large set of values to fine tune the loop gain according to the failure modes, the airplane state and control mode. The interfaces between the DCPU and the CIU to support this function involve:

1. The digital transmission of a word to select the loop gain, and

2. The digital or analog transmission of the position command.

The digital interface can be implemented either by using the DCPU backplane bus or a point-to-point digital parallel

interface. A serial point-to-point interface is less desirable than the parallel interface because the nature of the task requires parallel data.

The analog interface will be implemented with standard D/A and A/D converters. A wide variety of off-the-shelf components exist which can be used for these tasks. The accuracy achievable with 8 bit converters is ±1%. The accuracy achievable with 12 bit converters is ±.05%, which certainly exceeds the needs. The accuracy requirements will be determined during the design phase of this program.

TEST CONTROL TERMINAL (TCT)

The Test Control Terminal (TCT) performs three major functions in the IRAS lab. It is the IRAS control station; a software work station and host communications interface; and it emulates the functions of the Digital Pilot Display (DPD). The TCT communicates with the DCPU, the Arbitrator, the EC and the host development environment via RS-232 point-to-point serial links. The test engineer controls the entire IRAS environment from the TCT. He can select initial conditions; control program execution; direct the data gathering and recording processes; and trigger faults. The TCT also functions as a work station and intelligent terminal for host communications. Software for the EC and DCPU is stored on disks in the TCT, and loaded from these into the system.

The Digital Pilot Display (DPD) is implemented in the TCT. System status information and failure annunciations are transmitted directly from the EC to the TCT and displayed on the TCT.

The TCT must have sufficient hardware/software flexibility and availability to support future growth. One promising area of future research is the application of Artificial Intelligence concepts to maintenance and diagnostics procedures. Ground-based, expert system, maintenance and diagnostic programs could be developed and executed in the TCT.

After considering all the current functional requirements and future expansions, the IBM PC AT has been selected as the best system to implement the functions of the TCT. The IBM PC is the industry standard for medium sized microcomputers and a large number of hardware and software packages are available for that processor. In the area of Artificial Intelligence, the PC hosts both LISP and PROLOG, and there is a number of expert system development packages currently available for the PC. The computational power, execution speed and memory space of the IBM PC/AT meet or exceed the current and projected requirements of the TCT.

EMULATION COMPUTER (EC)

The primary function of the Emulation Computer (EC) is to emulate the Flight Control Computers (FCC), the diagnostic and maintenance computers, and the I/O interfaces between the FCC, ISCU and TCT. The EC houses the emulated Flight Control Computer (EFCC) software and the Emulated Diagnostic and Maintenance Computer (EDMC) software; and, it contains the necessary hardware for interfacing with the ISCU and the TCT.

The function of the EFCC software is to provide inputs to the ISCU which emulate those from Digital Flight Control Systems. The EFCC is initially configured as a synchronized quadraplex system and hosts a simple driver program for the DCPU which can be used by the test engineers to exercise the IRAS with prestored inputs such as: step, ramp and sine wave. The capability exists in the EFCC to implement dynamic, real-time simulations of fixed wing and rotary wing vehicles. This provides an environment to analyze the IRAS dynamic behavior, and the interactions with the vehicle dynamics during realistic flight maneuver. The EFCC also has the capability of emulating various FCC architectures, such as triplex, dual-dual, asynchronous, etc.

The EDMC software provides the on-line diagnostic environment of advanced DFBW avionics systems by interacting with the DCPU diagnostic software. The EDMC is initially configured to gather data in real-time for off-line processing. Status and failure information are stored in the EC memory during test execution, then downloaded to TCT for later evaluation. The EDMC can also initiate in-line diagnostics of a failed DCPU by triggering the execution of self-test software resident in the DCPU and it can monitor the results. More complex EDMC software can actively participate in the diagnostic test by sending a stimulus to the DCPU in the form of bit patterns and interpreting the DCPU reactions.

The EDMC is initially configured to provide the real-time functions performed by the Pilot Status Panel of the TAFCOS actuator. This is accomplished by the EDMC transmitting to the TCT in the number of failed channels: 0, 1, 2 or more than 2. The TCT utilizes this information to drive the appropriate CRT frames which simulates the cockpit pilot displays. The interfaces between the EC and other IRAS elements are described in another section of this report.

As in the case of the Arbitrator, and for the same reasons, the EC functions will be implemented in a dedicated 68000 processor.

FCC/DCPU INTERFACE

The following tasks were performed to develop the functional requirements for the FCC/DCPU interface.

1) A benchmark was developed to evaluate candidate interfaces.

2) A preliminary analysis of five interfaces was conducted.

The benchmark can be used to assess the relative performance of various interfaces. Absolute performance criteria could not be established in this analysis, because of their dependence on specific flight control system implementations. As an example, the required bus transmission time can be a very important factor in an implementation where the bus is operating at or near full capacity. Conversely, it can be a negligible factor if the bus is operating at a small fraction of the maximum transmission capacity.

For the purpose of establishing the system and message benchmark structure, the following was assumed:

1) The FCC and DCPU architecture are quad.

2) The time critical communications from each FCC to the DCPUs consist of four actuator commands, one for each control axis. Each command requires the transmission of two bytes.

3) The time critical communications from each DCPU to the FCCs consist of four LVDT position feedbacks, one for each of the four FCC channels. Each feedback signal requires the transmission of two bytes.

4) Cross-strapping and non cross-strapping configurations are used between FCCs and DCPUs.

For the purpose of establishing the interface benchmark structure, the following was assumed:

1) In the case of bus interfaces, for a non-cross-strapped configuration, each FCC has a bus controller and each channel of each DCPU has a bus terminal. In a cross-strapped configuration, each FCC must also have a bus terminal interface.

2) In the case of dedicated interfaces and no cross-strapping, each FCC is directly connected to the corresponding channel of each DCPU.

3) In the case of dedicated interfaces with cross-strapping, each FCC is directly connected to all channels of each DCPU.

The resultant data flow structures are shown in Figure 25 and 26 for non cross-strapped and cross-strapped configurations, respectively.

Preliminary analysis

Five interfaces have been evaluated. They are:

1) MIL-STD-1553B serial date bus in broadcast and standard mode.

2) Ethernet serial data bus

3) IEEE-488 parallel data bus

4) RS-232 serial point-to-point connection

5) RS-422 serial point-to-point connection

The broadcast mode of the 1553 bus has a different message structure than those discussed earlier. In this mode, the FCC transmits a message, and all terminals receive it simultaneously. This structure is shown in Figure 27 for both cross-strapped and non cross-strapped configurations.

Each interface was evaluated to determine the amount of transmission time required to transmit all the FCC to DCPU and DCPU to FCC messages. The detailed results are shown in Table 3 and are summarized in Figure 25 for non cross-strap configuration, Figure 26 for the cross-strap configuration and Figure 27 for 1533 broadcast mode.

The major conclusions of this analysis are:

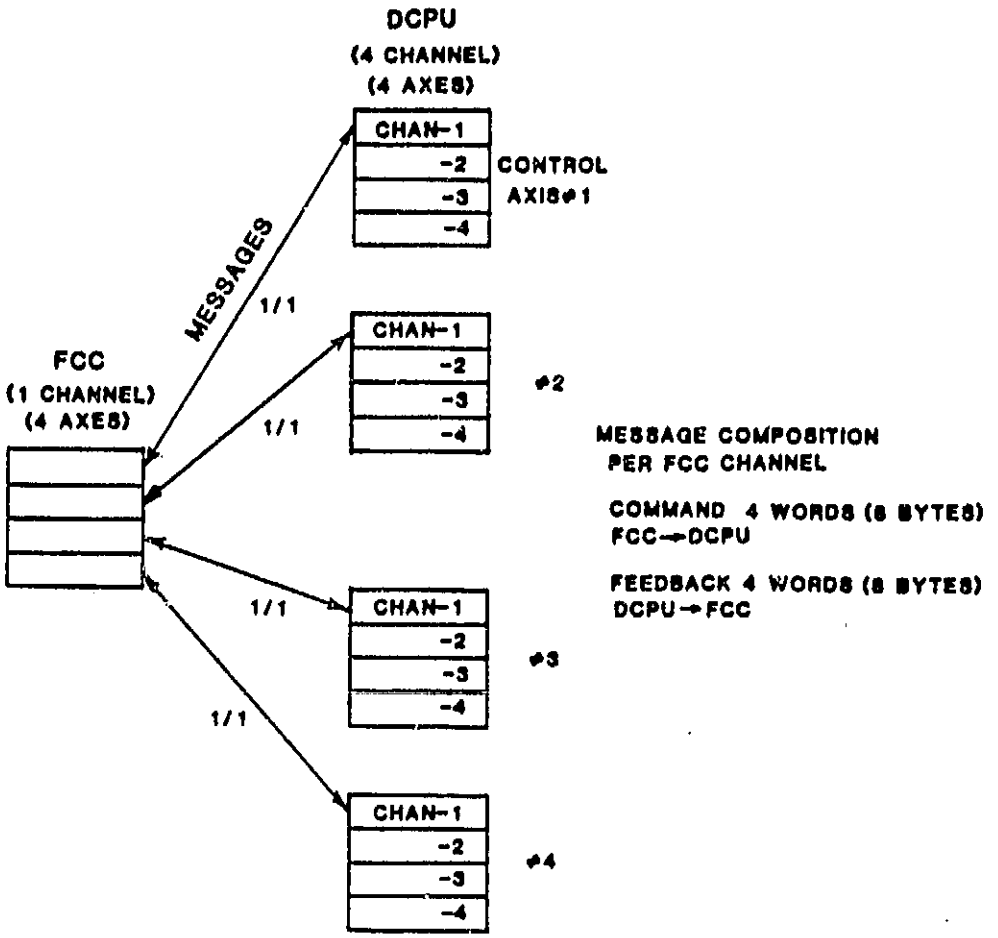
1) The bus overhead is significant for all the evaluated bus interfaces. The minimum value was obtained for the IEEE-488 and that is 60% of the total transmission time.

2) None of the bus structures studied have adequate performance to support 4-axis cross-strapped configurations.

3) The 1553 bus does not perform adequately even in the case of non cross-strapped configurations.

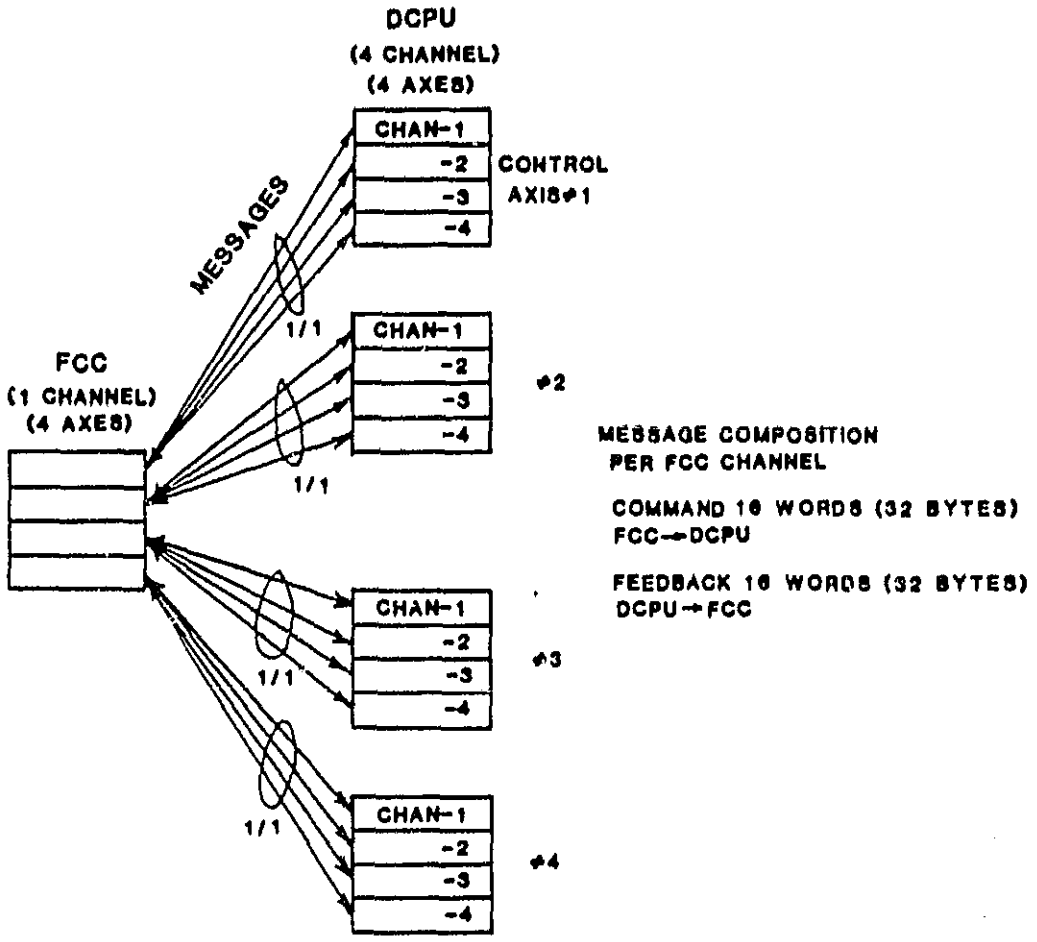
4) Ethernet is not appropriate for flight critical applications because the mechanism to resolve bus contention does not result in predictable transmission times.

5) The only interface which might be marginally acceptable is the RS-422 serial point-to-point connection.



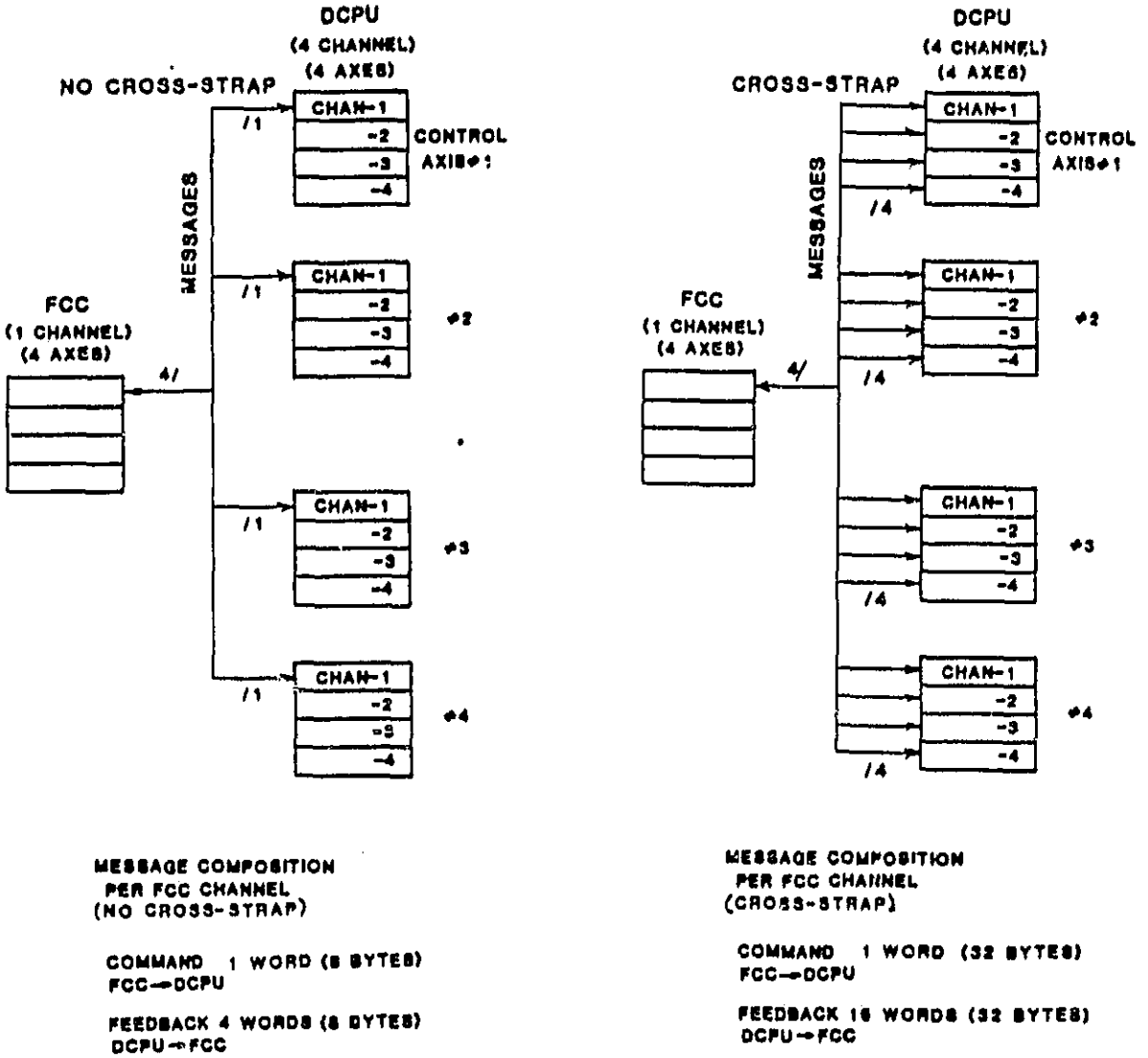
CONFIGURATION	TRANSMISSION			% OVERHEAD
	RATE MB/SEC	TOTAL BYTES	TOTAL TIME msec	
1553 NON-BROADCAST	.1	320	3.2	80
ETHERNET	1.25	1088	.87	94
IEEE-488	.25	160	.64	60
RS-232 (POINT TO POINT)	.001	64	64	-
RS-422 (POINT TO POINT)	.125	64	.256	-

FIGURE 25
FCC/DCPU DATA FLOW, NO CROSS-STRAP CONFIGURATION, 4 CHANNEL TRANSMISSION



CONFIGURATION	TRANSMISSION			% OVERHEAD
	RATE MB/SEC	TOTAL BYTES	TOTAL TIME msec	
1553 NON-BROADCAST	.1	1280	12.8	80
ETHERNET	1.25	4352	3.48	94
IEEE-488	.25	640	2.56	60
RS-232 (POINT TO POINT)	.001	256	256	-
RS-422 (POINT TO POINT)	.125	256	1.02	-

FIGURE 26
FCC/DCPU DATA FLOW, CROSS-STRAP CONFIGURATION, 4 CHANNEL TRANSMISSION



CONFIGURATION	TRANSMISSION			% OVERHEAD
	RATE MB/SEC	TOTAL BYTES	TOTAL TIME msec	
NO CROSS-STRAP	.1	224	2.24	71
CROSS-STRAP	.1	704	7.04	78

FIGURE 27

FCC/DCPU DATA FLOW, 1553 BROADCAST, CROSS-STRAP VS. NON CROSS-STRAP CONFIGURATION, 4 CHANNEL TRANSMISSION

TABLE 3 FCC/DCPU DATA FLOW

	CONFIGURATION	CMD	# of MESS.	# of BYTES	LVDT FB	# of MESS.	# of BYTES	OVERHEAD BYTES/ MESS.	OVER-HEAD BYTES	TOTAL BYTES /FCC	TOTAL BYTES 4CHAN.	XMISSION TIME msec	% OVER-HEAD
NO CROSS-STRAP	1553 NON-BROADCAST	4	4	8	4	4	8	8	64	80	320	3.2	80
	1553 BROADCAST	4	1	8	4	4	8	8	40	56	224	2.24	71
	ETHERNET	4	4	8	4	4	8	32	256	272	1088	.87	94
	IEEE-488	4	4	8	4	4	8	3	24	40	150	.64	60
	RS-232	4	4	8	4	4	8	0	0	16	64	64	0
	RS-422	4	4	8	4	4	8	0	0	16	64	.512	0
CROSS-STRAP	1553 NON-BROADCAST	16	16	32	16	16	32	8	256	320	1280	12.8	80
	1553 BROADCAST	4	1	8	16	16	32	8	136	176	704	7.04	78
	ETHERNET	16	16	32	16	16	32	32	1024	1088	4352	3.48	94
	IEEE-488	16	16	32	16	16	32	3	96	160	640	2.56	60
	RS-232	16	16	32	16	16	32	0	0	64	256	256	0
	RS-422	16	16	32	16	16	32	0	0	64	256	2.048	0

TRANSMISSION RATE – MB/SEC

1553= .1 ETHERNET= 1.25 IEEE-488=.25 RS-232=.001 RS-422= .125

In the succeeding phases of this program, additional bus protocols will be analyzed, which might be better suited for this application. The new DATAC bus, under development at Boeing, looks very promising.

SOFTWARE REQUIREMENTS

The software needed to support IRAS can be functionally partitioned into three areas as outlined in Figure 28.

- o Application
- o Executive
- o Utility

The application software includes the servoactuator control; the failure detection and isolation algorithms; the reconfiguration management strategies; the simulation of the FCC; and the pilot displays. The executive software provides executive functions such as: scheduling of tasks; interchannel and intrachannel synchronization; I/O interfaces and device drivers; and interrupt servicing. The utility software includes program execution control; fault insertion mechanism; data collection and analysis; program execution monitoring; and, software development tools.

All the application software, a considerable percentage of the executive software and a small percentage of the utility software will be developed exclusively for IRAS. The rest will be acquired from software houses and computer manufacturers in the form of standard software packages. Examples of this type are language processors, debuggers, and operating systems. The application software is hosted primarily in the DCPU processors, except that the FCC simulation is hosted in the EFCC, and the pilot display function hosted in the TCT.

APPLICATION SOFTWARE

The design of the application software will reflect the requirements of being "user friendly" to the maximum possible extent. In fact, the IRAS must be flexible and the flexibility and reconfiguration capabilities are, for the most part, achieved through changes of the application software. For this reason, the following design guideline will apply:

1. The software will be highly modularized. The module size will be limited to 50 lines of code or less and the modules will be arranged in a hierarchical manner.

2. The software will be coded in a high order language (HOL) using only structured constructs. As an example, unconditional "go to" to transfer control from one module to another or within the same module will not be allowed.

APPLICATION AREA	EXECUTIVE	UTILITIES
SERVOCONTROL LOOP	SCHEDULING & RATES	SIMULATION CONTROL
FAILURE DETECTION AND ISOLATION	SYNCHRONIZATION	FAULT INSERTION
RECONFIGURATION	I/O AND INTERFACES	DATA COLLECTION & ANALYSIS
FCC SIMULATION	DEVICE DRIVERS	EXECUTION MONITORING
PILOT DISPLAY	INTERRUPTS	DEVELOPMENT ENVIRONMENT

FIGURE 28
IRAS SOFTWARE STRUCTURE

3. Each module will include a preamble to explain the function of the module and the module interfaces. It will include a definition of each input and output variable including: origin, destination, physical units, and range of values. A section of each module will contain the numerical values of all constants and all "initial conditions", as well as the execution requirements such as the execution sequence and the required minimum execution rate.

4. Comment statements will be dispersed throughout the code to facilitate the understanding of each statement or group of statements.

EXECUTIVE SOFTWARE

The requirements for the executive software reflect purposes and user requirements quite different from those of the application software. The executive software provides the structure needed for the proper and timely execution of the application software. The most critical tasks performed by the executive software are:

1. Provide a framework which guarantees the execution of each application package within the required frame time. If different algorithms require different execution rates, then a structure which includes fast and slow frame times must be provided. Within this structure, an orderly progression of task executions must be maintained (input/processing/output) to minimize computational lags.

2. Provide several levels of synchronization such as the FCCs, the DCPUs, and the FCCs with DCPUs.

3. Support the digital interfaces between the EFCC and the DCPUs. These digital interfaces are implemented either by a bus architecture or a direct link.

4. Support the digital interfaces within the DCPUs and the Arbitrator. These interfaces are all implemented by the backplane common bus.

5. Support the interfaces between the TCT and the IRAS. These interfaces are implemented through an RS-232 digital serial link.

6. Support the driver of any special device such as D/A and A/D converters.

7. Provide the modules needed to service clock and I/O interrupts.

The executive software can support a wide variety of applications and IRAS configurations and the only foreseeable future changes are those which might be required to support hardware extensions. The user is rarely concerned with the implementation intricacy of the executives; since his only interest is in the capability that they provide. Therefore, the executive software modules are not subjected to a high level of visibility which was the case for the software application modules. For this reason, and because the executive is much closer to the hardware than the application software, assembly code implementations are acceptable and indeed they might be the only available solution in many situations. Visibility will be provided in those user interface areas such as: setting of execution iteration rate, input/output tables, etc.

UTILITY SOFTWARE

Utility software provides the following functions:

1. The overall control of the closed loop, real-time, simulation environment.

2. The capability of simulating, evaluating or directly inserting faults in the FCC output plane, DCPU, Arbitrator, and CIU.

3. Some general capability of collecting data in real-time for later analysis.

4. The capability of monitoring the execution time of modules or a group of modules and generating an alarm when the maximum allowable time limits are exceeded.

5. The support of software development and verification processes.

6. The support of off-line data analysis functions.

As previously stated, most of the utility software programs and relative program documentation will be purchased from vendors. However, some configuration dependent functions will be developed exclusively for IRAS support. These functions include: the simulation control, fault insertion capability, and the real-time execution monitoring of the DCPU modules. The documentation provided with these programs will primarily consist of "user guide" directives with minimum documentation on implementation details.

SOFTWARE DEVELOPMENT ENVIRONMENT

Two software development environments are needed to support the IRAS system. The first supports the IBM personal computer architecture for the TCT and the second; the software development for the EC, DCPU, and Arbitrator.

The development environment for the IBM PC consists of the standard IBM Disk Operating System (DOS) and DOS utility functions. This core system will be augmented with the following packages:

1) FORTRAN, Pascal or "C" compiler;

2) TCT communications; and

3) Graphic capabilities.

The final decision between "FORTRAN", "Pascal", or "C" as the HOL for the TCT programs will be made during the design phase. All TCT communication links are implemented via a RS-232 serial interface including communications to the VAX and IRAS. The TCT graphic capabilities are needed to simulate pilot displays.

Two environments have been evaluated to support EC, DCPU and Arbitrator software developments: one resident in a dedicated 68000 processor and the other hosted in a VAX minicomputer. The VAX environment was selected primarily for the following reasons:

1) Better tool selection, quality and power.

2) Excellent customer support which is practically nonexistent for the 68000 environment.

3) Availability of integrated software development and engineering analysis tools.

The core of the VAX resident software development environment systems will consist of:

1) 68000 cross-compiler, cross-assembler and linker, and

2) 68000 debug monitor.

The debug monitor is needed to perform the functions of program downloading, reading and writing contents of registers, and memory execution control.

Probably the most important element of the development environment is the HOL. A preliminary analysis has been completed to determine which of four promising languages is best suited for the IRAS application. The four languages are: FORTRAN, Pascal, "C" and Ada. The following was determined:

Fortran The strengths are:

1) The language is well understood by most engineers, even those without any formal computer science background;

2) Good compilers and cross-compilers are available for most processors; and

3) A large selection of engineering analysis tools is available.

The weaknesses are:

1) The language has poor control and data structures;

2) It does not support type checking; and

3) It does not support recursive or reentrant procedures.

Pascal The strengths are:

1) The language has good control and data structures;

2) It supports type checking;

3) It supports recursive procedure; and

4) Compilers are available for most processors.

The weaknesses are:

1) It does not support separate compilation modules. Each module can not be compiled separately and then linked with previously compiled modules. As a result, a change in a single module requires the recompilation of all modules.

2) It does not support external procedures; and

3) It does not allow the undesirable but sometime necessary mixing of HOL and assembly statements.

(Some of the Pascal weaknesses have been removed in dialects of the standard language; there are, however, distinct disadvantages when a non-standard HOL is used.)

"C" The strengths are:

1) The language has good control and data structures;

2) It supports low level programming features such as logical AND/OR at a bit level and arithmetic shifts; and,

3) It allows direct access of I/O registers.

The weaknesses are:

1) The knowledge of the language is not as widespread among engineers as FORTRAN and Pascal.

Ada The strengths are many. However, Ada is not a mature language, and no known validated production compiler exists for the 68000.

Based on the preliminary analysis, "C" clearly is the best language. It is powerful and has features which make it very well suited for a real-time, embedded application such as, the logical statements at the bit level and direct addressability of I/O registers. The language enjoys a rapidly growing popularity, and many tools and advanced environments, such as UNIX, are built around it. Unless some hidden incompatability with the overall requirements of IRAS is detected, "C" will be the HOL language for that system.

SOFTWARE DEVELOPMENT PROCESS REQUIREMENTS

The software development process is structured in three phases: design, coding and verification/validation.

In the design phase, a hierarchical, top-down, module structure will be developed first, followed by the specification of each module. This orderly process is necessary to comply with: 1) the IRAS flexibility requirements, and ease of use and modification; 2) all the functional requirements including critical performance functions such as hardware interfaces and iteration rates; and 3) the requirements for a minimum module complexity. In this phase, testing procedures will be established for each module and to the maximum possible extent, for module integration.

The software coding process includes: the development of a top-down module development plan; module coding; and module integration. The module development plan is needed to provide an orderly progression of module development and integration consistent with the hierarchical module organization developed in the previous phase. The module development plan will also generate the testing procedure for module integration which could not be developed in the design phase due to the lack of detailed

definition of some structures. The complete integration test procedures will include tests of the interfaces, timing synchronization mechanisms and execution sequences. Each module will be coded in the order specified by the module development plan.

The verification/validation process will have the same top-down structure as the development process. This will guarantee a "most critical--first tested" approach. The top-down approach will require the development of "test stubs" which simulate the interface with lower level modules, while testing high level modules. The verification process will mainly consist of module and integration testing. It will be accomplished by following the test procedures established during the design phase and the initial phase of the coding process. Each module and group of integrated modules will be statically and dynamically tested. Module testing will concentrate on the module specification, while integration testing will primarily address interface and synchronization issues. Formal and informal design coding reviews will be conducted to assure compliance with system requirements, module definitions, and design and coding guidelines. All the test procedures will be described, and the results discussed in a dedicated document which will also be used as the basis for the system acceptance.

EMBEDDED SOFTWARE DEVELOPMENT SCENARIO

IRAS embedded software will be developed in a Digital Equipment Corporation VAX computer system using cross-development tools. The resulting execution module will then be downloaded to the IRAS through the TCT, for execution in the embedded computers.

The first task of the development process will be the coding of a source module using the VAX system editor. The completed source module will then be compiled using the microprocessor HOL cross-compiler which will result in either a relocatable object module or on an assembly language module. If an assembly language module is generated, then it must be assembled with the microprocessor cross-assembler to produce the relocatable object module.

The next task is linking the relocatable object modules with the HOL runtime library and the IRAS executive software using the microprocessor cross-linker. The resulting executable module will then be downloaded from the VAX to the TCT using a standard communication package. Then it can be stored on either a floppy or hard disk drive and downloaded from the TCT to the embedded computers for execution. Program execution and monitoring is controlled through the TCT.

APPENDIX A

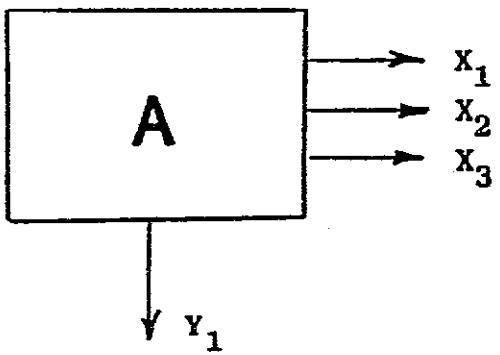
RELIABILITY ANALYSIS

A preliminary reliability analysis of several IRAS configurations was completed for this study. A graphic manual documentation technique and an advanced computer program have been used to support this analysis and are briefly discussed here for completeness and clarity.

The documentation technique is based on reliability diagrams which show the interdependency existing among the major components of each system. These reliability diagrams closely resemble data flow diagrams and can be used at different levels of abstraction and detail.

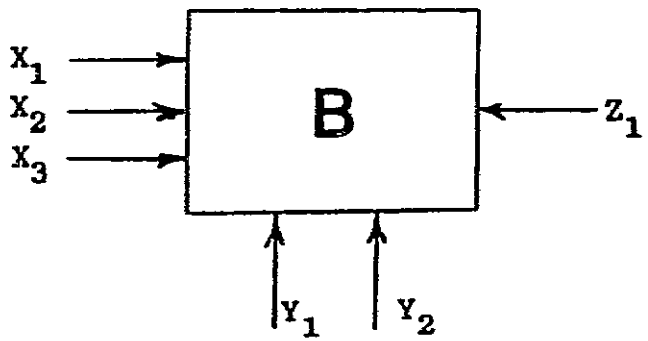
The elements of the reliability diagrams are boxes labeled with a letter and arrows ⟶ as shown in Figure A-1. Each box is a function or component of the system. Boxes using the same label represent redundant components which perform identical functions using identical sets of inputs to produce an identical set of outputs. Boxes with unique labels represent single point failure functions or components for which redundancy is not provided. The arrows represent the inputs to, and the outputs of, each box. Arrows entering the box from the same side represent identical inputs. Inputs of different types are represented by arrows entering the box from different sides. The failure of a function occurs when all the identically labeled boxes representing or performing that function fail. Each box has a predicted failure rate and, therefore, has an inherent probability of failure. A box also fails if all the inputs to one side fail, indicating that all the inputs of a certain type have failed. If a box fails, its output fails as well, and all the boxes which depend upon that output also fail unless redundant signals are provided. Terminal boxes (such as "box B" in Figure A-1) have no output and represent the ultimate functions of the system or the system components which perform those ultimate functions.

The computer program used to support this analysis is a powerful tool hosted in HR Textron's computer facility for scientific applications. The program is capable of analyzing highly redundant configurations and the effects of transient faults. The program takes into account that the loss of certain components causes the loss of other components (component dependency) and has some capability for handling functional redundancy among different components. The program was validated in the HR Textron, Valencia Facility, both by comparing the output with those generated by other models and by hand computation of a wide variety of cases.



IF A FAILS THEN

X_1 AND X_2 AND X_3 AND Y_1 FAIL



B FAILS IF

X_1 AND X_2 AND X_3 FAIL OR

Y_1 AND Y_2 FAIL OR

Z_1 FAILS OR

B FAILS

FIGURE A-1 RELIABILITY DIAGRAM SYMBOLS

APPENDIX B

The following is a summary description of the implementation mechanizations and protocols of the bus architectures which have been exercised for the EFCC/DCPU interfaces.

MIL-STD-1553B

The MIL-STD-1553B defines a 1 MHZ, bit serial, time division command/response multiplex data bus for use in integration of aircraft subsystems. Bus nodes are either a remote terminal or a bus controller. A maximum of 31 remote terminals and one bus controller are supported by the standard.

All communications on the 1553B data bus are initiated by the bus controller. Data transfers can be from the bus controller to a remote terminal; from a remote terminal to the bus controller; a remote terminal to another remote terminal; the bus controller to all remote terminals (broadcast mode); and from a remote terminal to all other remote terminals (broadcast mode). For a 4 channel, 4 axis, non-cross-strapped configuration, a total of 32 messages are transmitted: 16 command messages from the FCC to the ISCU and 16 feedback signals from the ISCU to the FCC. The transmission time is 100 μsec. per message, for a remote terminal to remote terminal transmission, and the constant total transmission time is 3.2 msec. In the case of a quad redundant bus structure with one 1553B bus per channel, the transmission time is reduced to slightly more than 1 millisecond. The transmission time can further be reduced if the FCC also acts as the bus controller. In this configuration, the total transmission time for each bus is 576 μsec.

For cross-strapped configurations using a single 1553B data bus, the total transmission involves 128 messages or a total transmission time of 12.8 msec. Using a quad 1553B bus structure with each of the FCCs acting as the controller for one bus, reduces the transmission time to 3.0 milliseconds.

ETHERNET

Ethernet, also known as IEEE-802.3, has become the de facto standard for an office environment local area network. Ethernet uses a 10 MHZ bit serial baseband bus structure for node interconnection, and resolves bus contention through the use of carrier sensing multiple-access with collision detection (CSMA/CD). When a collision occurs, all nodes must cease transmitting and wait a random amount of time before attempting to regain control of the bus.

The minimum ethernet packet size, containing 46 data bytes, is 72 bytes and 67.2 microseconds long. The maximum packet size, containing 1500 data bytes, is 1526 bytes and 1.2 milliseconds long.

Ethernet, or any other bus protocol using CSMA/CD for bus arbitration, is inappropriate for use in an aircraft control system. The primary deficiency of CSMA/CD is that it assumes collisions will occur infrequently and thus will have very little effect on performance. In an aircraft flight control system, however, collisions are very likely to occur and at the most critical time: when the position commands are transmitted to the servo control processors.

IEEE-488

The IEEE-488 bus, also known as the General Purpose Interface Bus (GPIB), was developed by the Hewlett-Packard Corporation for interfacing laboratory instruments to data acquisition and control systems. The interface has 24 lines; 8 bidirectional data bus lines, 3 data byte control lines, 5 general interface management lines, and 8 ground lines. A device on the GPIB may perform three types of functions; talker, listener, and controller. As the names imply, a talker is a device that transmits data on the bus; a listener, a device that receives data from the bus; and a controller, one that controls which device is allowed to talk and which device or devices are to listen. The GPIB protocol uses a large amount of handshaking and does not appear to be intended for applications that have frequent changes of the talking device.

There are several attributes of the GPIB that make it, or any derivative of it, undesirable for a flight system:

1. The bus requires too many lines (24), which has a negative impact on reliability and weight.

2. The maximum length of the bus is limited to 20 meters.

3. The bus controller is a single point failure.

RS-232

RS-232 is the standard developed by the Electrical Industry Association (EIA) in cooperation with the Bell System, for interfacing Data Terminal Equipment (e.g., computers and CRT's) with Data Communication Equipment (i.e., modems). RS-232 implements a bit serial bidirectional point to point connection capable of supporting up to 9.6k baud over a 50 foot distance. RS-232 is not supported by a standard protocol and this is left to the particular application.

RS-422

The RS-422 standard was developed in response to the need for a communication protocol supporting longer distances and higher transmission speed than was provided by existing standards. It defines a bit serial, point-to-point interface capable of supporting up to a 1 megabit transmission rate over 400 feet or up to 100k bits per second over 4000 feet. Like the RS-232 standard, the RS-422 does not define a communication protocol.

ARINC-429

The ARINC-429 standard defines a bit serial, multidrop, unidirectional bus capable of transmitting at 100k or 14k bits per second. It is designed to provide aircraft sensor input data to flight control computers and therefore supports only one transmitter (the sensor), and several receivers (the FCCs). The unidirectional feature of the ARINC-429 makes it unacceptable for IRAS, which requires bidirectional communication.

ABBREVIATIONS

A/D	-	Analog To Digital
ADOCS	-	Advanced Digital/Optical Control System
AFCS	-	Advanced Flight Control System
AI	-	Artificial Intelligence
CIS	-	Cubic Inches Per Second
CIU	-	Control Interface Unit
CPU	-	Control Processing Unit
CRT	-	Cathode Ray Tube
CSMA/CD	-	Carrier Sensing Multiple Access with Collision Detection
D/A	-	Digital to Analog
DATAC	-	Digital Autonomous Terminal Access Communication
DCPU	-	Digital Command Processing Unit
DOS	-	Disk Operating System
DPD	-	Digital Pilot Display
DSI	-	Data System Interface
EC	-	Emulation Computer
EFCC	-	Emulated Flight Control Computer
EIA	-	Electronic Industries Association
EMDC	-	Emulated Diagnostic and Maintenance Computer
ESCU	-	Electronic Servoactuator Control Unit
FAIL OP_2	-	Fail Operational
FAIL OP^2	-	Fail Operational/Fail Operational
FCC	-	Flight Control Computer
FS	-	Full Scale
GPIB	-	General Purpose Interface Bus
HOL	-	Higher Order Language
I/O	-	Input/Output
IRAS	-	Intelligent Redundant Actuation System
ISCU	-	Intelligent Servoactuator Control Unit
LVDT	-	Linear Variable Displacement Transducer
ma	-	milliampere
MSEC	-	Millisecond
PC	-	Personal Computer
Quad	-	Quadruplex
μSEC	-	Microsecond
V	-	Volt

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR - 177366		

4. Title and Subtitle	5. Report Date
IRAS Requirements and Preliminary System Design	September 1985
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
P. de Feo, J. Harris, L. J. Geiger	HR 73600253
	10. Work Unit No. T - 3676
9. Performing Organization Name and Address	11. Contract or Grant No.
HR Textron Inc. 2485 McCabe Way Irvine, California 92714	NAS2 - 12081
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Contractor Report
National Aeronautics and Space Administration Washington D.C. 20546	14. Sponsoring Agency Code 307 - 04 - 1

15. Supplementary Notes

Point of Contact: Technical Monitor, K. C. Shih, M/S 210-5
Ames Research Center, Moffett Field, CA 94035
(415) 694-6687 or FTS 464-6687

16. Abstract

Several redundant actuation system configurations have been designed and demonstrated to satisfy the stringent operational requirements of advanced flight control systems. However, this has been accomplished largely through "brute force" hardware redundancy, resulting in significantly increased computational requirements on the flight control computers which perform the failure analysis and reconfiguration management. Modern technology now provides powerful, low-cost microprocessors which are effective in performing failure isolation and configuration management at the local actuator level. One such concept, called an Intelligent Redundant Actuation System (IRAS), significantly reduces the flight control computer requirements and performs the local tasks more comprehensively than previously feasible. This document outlines the requirements and preliminary design of an experimental laboratory system capable of demonstrating the concept and sufficiently flexible to explore a variety of configurations.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Redundant Actuation Systems, Digital Servoactuator Controllers, Criticality, Microprocessor, Reliability, Fault-tolerant Systems	Unclassified - Unlimited STAR Category 05

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	71	

*For sale by the National Technical Information Service, Springfield, Virginia 22161